Requested Patent: JP6202996A

Title:

EXPANSION FUNCTION OF SUBSCRIBER-DISTRIBUTED TWO-PHASE COMMIT PROTOCOL ;

Abstracted Patent: JP6202996 ;

Publication Date: 1994-07-22 ;

Inventor(s):

BROCKMEYER ROGER L;; DIEVENDORFF RICHARD;; HOUSE DANIEL E;; JENNER EARLE H;; LABELLE MARGARET K;; MALL MICHAEL G;; SILEN STUART L ;

Applicant(s): INTERNATL BUSINESS MACH CORP lt;IBMgt; ;

Application Number: JP19930202200 19930816 ;

Priority Number(s): ;

IPC Classification: G06F15/16; G06F13/00 ;

Equivalents: JP2675968B2 ;

ABSTRACT:

PURPOSE: To provide an expansion function of two-phase commit protocol which attains the subscription of distributed subscribers between physically separated agents without relying upon the communication mechanism used by data processing systems.

CONSTITUTION: A special processing state called first-phase (EPOP) which can distribute coordinator functions by using an arbitrary communication mechanism is added to a two-phase commit protocol. A subscriber sends a two-phase commit protocol sequence to a distributed system in a special stage in which the subscriber can acquire a control right. The communication mechanism is used so that the mechanism can become part of a distributed coordinator. The coordinator itself does not require the knowledge about the other systems. The special processing system is validated by operating system service called first-phase terminating process enable (EEPOEP). Because of the EEPOEP, the expansion function of the two-phase commit protocol can be used on an executing-side system.

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平6－202996

(43)公開日　平成６年(1994)７月22日

(51)Int.Cl.[5]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	15/16	３７０　Ｍ	7429－5L		
	13/00	３５５	7368－5B		

審査請求　有　　請求項の数12　（全 29 頁）

(21)出願番号　特願平5－202200

(22)出願日　平成５年(1993)８月16日

(31)優先権主張番号　９３２８３５

(32)優先日　1992年８月20日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　ロジャー・ライル・ブロックマイヤー
アメリカ合衆国95139、カリフォルニア州　サン・ホセ、スカウヘイガン・コート　148

(74)代理人　弁理士　合田　潔　（外３名）

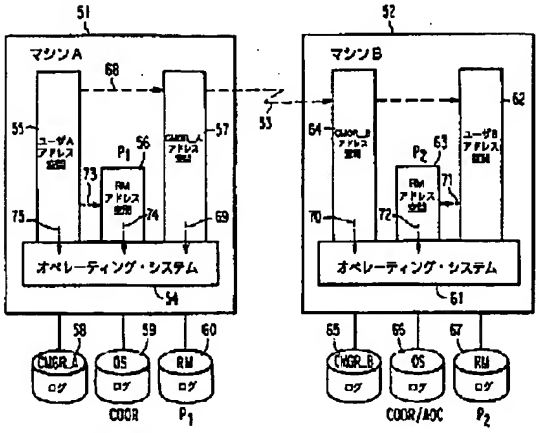
最終頁に続く

(54)【発明の名称】　加入者分散２相コミット・プロトコルの拡張機能

(57)【要約】　（修正有）

【目的】　データ処理システムで用いられている通信メカニズムに依存せずに、物理的に離れたエージェント間で分散加入を可能にする２相コミット・プロトコルの拡張機能を提供する。

【構成】　任意の通信メカニズムを使用してコーディネータ機能を分散させることができる第１相終了処理（ＥＰＯＰ）と呼ばれる特別処理ステージが２相コミット・プロトコルに追加される。加入者が制御権を得られる特別ステージで、２相コミット・プロトコル・シーケンスを分散システムに送る。通信メカニズムは、それが分散コーディネータの一部になるように用いられる。コーディネータそれ自体は他のシステムの知識を要しない。特別処理ステージは、第１相終了処理イネーブル（ＥＥＰＯＥＰ）と呼ばれるオペレーティング・システム・サービスによって有効になる。ＥＥＰＯＥＰにより、２相コミット・プロトコルの拡張機能が実行側システム上で使用できる。



1

【特許請求の範囲】

【請求項1】分散データ処理システムにおける更新調整方法であって、

上記分散データ処理システム内の第1システム上で第1相終了処理を有効にし、上記第1システム上の第1コーディネータ機能が、上記分散データ処理システム内の第2システム上の第2コーディネータ機能のエージェントになるようにするステップと、

上記第1システム上のすべての加入者からの投票がすべて受信された時に、第1システムの投票の第1相終了レコードを上記第1コーディネータがログに書込むステップと、

上記書込みに応答して制御を上記第1相終了処理に引渡すステップと、

を含む更新調整方法。

【請求項2】準備のための上記投票が、加入者が作業単位の変更をCOMMITする準備が出来たことを示すYES、加入者が作業単位の変更をバックアウトしようとすることを示すNO、または加入者が作業単位に継続して関わることを望み、作業単位の最終決定には影響を与えようとしないことを示すABSTAINを含む、請求項1記載の分散データ処理システムにおける更新調整方法。

【請求項3】少なくとも第1オペレーティング・システムを実行する第1マシンと、第2オペレーティング・システムを実行する第2マシンとを含む分散データ処理システムにおける更新調整方法であって、

上記第1オペレーティング・システムが、上記第2マシン上の第2通信マネジャと通信する第1通信マネジャを含む上記第1マシン上の加入者に準備信号を送り、上記第1通信マネジャが上記準備信号を上記第2通信マネジャに送るステップと、

上記第2通信マネジャが、上記第1通信マネジャからの準備信号に応答して、上記第2オペレーティング・システムに対して第1相終了イネーブル・コールを生成し、

上記第2オペレーティング・システムが上記第1オペレーティング・システムのコーディネータ機能のエージェントになるようにするステップと、

上記第2オペレーティング・システムが、上記第2通信マネジャを含む上記第2マシン上の加入者に準備信号を送るステップと、

上記第2マシン上の上記加入者から投票を受信するステップと、

第1相終了処理により、上記第1通信マネジャに上記投票を送る上記第2通信マネジャに制御を渡すステップと、

を含む更新調整方法。

【請求項4】準備に応答する上記投票が、加入者が作業単位の変更をCOMMITする準備が出来たことを示すYES、加入者が作業単位の変更をバックアウトしようとすることを示すNO、または加入者が作業単位に継続して関わることを望み、作業単位の最終決定には影響を与えようとしないことを示すABSTAINを含む、請求項3記載の分散データ処理システムにおける更新調整方法。

2

【請求項5】上記第2マシン上の上記加入者からの上記投票の少なくとも一部がYESであり、上記第2オペレーティング・システムが、上記第2マシン上の上記加入者による投票の第1相終了レコードを第2マシン・ログに書込むステップを含み、上記第2通信マネジャに制御を引渡す上記ステップが、上記第2マシンログへの書込み後に第1相終了処理によって実行される、請求項4記載の分散データ処理システムにおける更新調整方法。

【請求項6】上記第2マシン上の加入者による上記投票のすべてがFORGETまたはABSTAINであり、

上記第1オペレーティング・システムが、上記第1通信マネジャによって上記第2通信マネジャから受信された投票を含むすべての投票を上記第1マシン上の加入者から受信して、アトミック・インスタント・レコードを第1マシン・ログに書込むステップと、

上記第1オペレーティング・システムが、上記投票の結果によって決定された実行信号を上記第1マシン上の加入者に送り、上記第2マシン上の上記加入者による上記FORGETまたはABSTAINの投票により上記第2通信マネジャには実行信号が送られないステップと、

を含む請求項4記載の分散データ処理システムにおける更新調整方法。

【請求項7】上記第1マシン上の加入者による準備のための投票がすべてFORGETまたはABSTAINであり、

上記第2オペレーティング・システムが、上記第2マシン上の上記加入者による投票の第1相終了レコードを第2マシン・ログに書込むステップと、

上記第2マシン・ログへの書込みの後、上記第1通信マネジャに上記投票を送る上記第2通信マネジャに制御を渡すステップと、

上記第1オペレーティング・システムが、上記第1通信マネジャによって上記第2通信マネジャから受信された投票を受信して、アトミック・インスタント・レコードを第1マシン・ログに書込むステップと、

上記第1オペレーティング・システムが、FORGETまたはABSTAINを投票した、上記第1マシン上の加入者には実行信号を送らず、上記第1通信マネジャに送り、上記第1通信マネジャが上記第2通信マネジャに上記実行信号を送るステップと、

上記第2オペレーティング・システムが、上記第2通信マネジャによって受信された上記実行信号に応答して上記第2マシン・ログに実行レコードを書込むステップと、

上記第2通信マネジャが、実行が完了したとの通知を上

記第１通信マネジャに送るステップと、
を含む更新調整方法。

【請求項８】上記第１オペレーティング・システムが、上記第１通信マネジャによって上記第２通信マネジャから受信された投票を含む上記第１マシン上の加入者からの投票をすべて受信して、アトミック・インスタント・レコードを第１マシン・ログに書込むステップと、
上記第１オペレーティング・システムが、上記投票の結果によって決定された実行信号を上記第１マシン上の加入者に送り、上記第１通信マネジャが上記実行信号を上記第２通信マネジャに送るステップと、
上記第２オペレーティング・システムが、上記第２通信マネジャによって受信された上記実行信号に応答して上記第２マシン・ログに実行レコードを書込むステップと、
実行が完了したことの通知を上記第２通信マネジャが上記第１通信マネジャに送るステップと、
を含む請求項４記載の分散データ処理システムにおける更新調整方法。

【請求項９】上記第２オペレーティング・システムが、上記第２マシン上の加入者から実行が完了したとの確認応答を受信して、上記第２マシン上の加入者に上記実行信号を送り、上記第２マシン・ログを更新し、上記第１オペレーティング・システムが、上記第１通信マネジャを含む上記第１マシン上の加入者から実行が完了したとの確認応答を受信して、上記第１マシン・ログを更新するステップを含む請求項８記載の分散データ処理システムにおける更新調整方法。

【請求項１０】拡張２相コミット・プロトコルを実現する分散データ処理システムであって、
第１オペレーティング・システムを実行し、第１通信マネジャを含む複数の第１加入者をサポートする少なくとも第１マシンと、
第２オペレーティング・システムを実行し、第２通信マネジャを含む複数の第２加入者をサポートする少なくとも第２マシンと、
上記第１及び第２の通信マネジャを相互接続する通信手段とを含み、
上記第１オペレーティング・システムが、上記第１マシン上の上記複数の第１加入者に準備信号を送り、上記第１通信マネジャが上記準備信号を上記第２通信マネジャに送り、上記第２通信マネジャが、上記第１通信マネジャからの準備信号に応答して、上記第２オペレーティング・システムに対して第１相終了イネーブル・コールを生成し、上記第２オペレーティング・システムが上記第１オペレーティング・システムのコーディネータ機能のエージェントになるようにし、上記第２オペレーティング・システムが、上記第２マシン上の加入者に準備信号を送り、上記第２オペレーティング・システムが、上記第２マシン上の上記加入者による投票の第１相終了レコードを第２マシン・ログに書込み、上記第２マシン・ログへの書込みの後、第１相終了処理により制御を上記第２通信マネジャに渡し、上記第２通信マネジャが上記投票を上記第１通信マネジャに送る、
分散データ処理システム。

【請求項１１】上記第１及び第２のマシンが、仮想マシン・オペレーティング・システム上で１つのコンピュータを実行する仮想マシンであり、上記第１及び第２のオペレーティング・システムが、上記仮想マシン・オペレーティング・システムのゲスト・オペレーティング・システムである、請求項１０記載の分散データ処理システム。

【請求項１２】上記第１及び第２のマシンが物理的に離れたコンピュータ上に実現され、上記通信装置が上記コンピュータを相互接続する通信リンクを含む、請求項１０記載の分散データ処理システム。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、一般的にはデータ処理システムに関し、特に加入者が遠隔加入者と相談するために決定プロセスを拡張できるように、或いは他のローカル操作が実行できるようにする２相コミット・プロトコルの拡張機能に関する。本発明は、通信マネジャを用いて、分散したコミット・スコープを実現できるシステムの開発に用いることができる。ユーザは分散可能であり、加入者も同様に分散可能である。

【０００２】

【従来の技術】標準的な２相コミット・プロトコルは、集中管理プロセス（すなわちコーディネータ）がサテライト・プロセス（すなわち加入者）によるデータ変更の結果の”硬化”を同期化することができる周知の方法である。２相とは”投票”相（すなわちＰＲＥＰＡＲＥ信号を加入者に送ること）と”実行”相（すなわちＣＯＭＭＩＴまたはＢＡＣＫＯＵＴ信号を加入者に送ること）である。

【０００３】加入者は、復元可能なデータを変更した時はコーディネータに通知しなければならない。データがコミットされる準備が出来ると（すなわちユーザがＣＯＭＭＩＴまたはＢＡＣＫＯＵＴを要求していると）、コーディネータは、２相コミット・プロトコルを使用して、すべての加入者が前へ進む（すなわちデータをコミットする）かまたは後へ進む（すなわちデータの変更をバックアウトする）ようにしなければならない。コーディネータは、全加入者の実行が同じになるようにしなければならない。ある加入者がコミットしてある加入者がバックアウトすることは全く許されない。現在の２相コミット・プロトコルでは、複数の加入者間でデータの保全性が維持されるのは、データに変更を加える前に全加入者の同意をとりつけることによる。

【０００４】２相コミット・プロトコルの詳細について

5

は下記の文献を参照されたい。

Cruz、R. 、Data Recovery in IBM Database 2、IBM Systems Journal、vol. 23、no. 2、1984。

Date、C. J. 、An Introduction to Database Systems、vol. II、Addison-Wesley (1983) 。

Gray、J. N. 、Notes on Data Base Operating Systems、IBM ReaserchReport RJ2188、Feb. 1978。

Haerder、T. 、and A. Reuter、Principles of Transaction-OrientedDatabase Recovery、ACM Computing Serveys、vol. 15、no. 4、December 1983。

Lindsay、B. 、and C. Mohan、Efficient Commit Protocols for the Tree ofProcesses Model of Distributed Transactions、IBM Research Report RJ3881、June 2、1983。

【０００５】２相コミット・プロトコルは分散データベースに応用されているが、その成果は限られる。従来技術では、分散データベースを扱うために集中管理とリニアの両方式の２相コミット・プロトコルが検討されている。一例として、C. J. Date、B. J. LindsayらによるNotes on Distributed Databases、IBM ResearchReport RJ2571、July 1979を参照されたい。この文献では323ページ以降で分散データベースに用いるコミット・プロトコルが解説されている。

【０００６】システム上の遠隔加入者がさまざまな通信媒体によって接続された分散データ処理システムに、２相コミット・プロトコルを応用する際には１つ問題がある。分散環境の場合、コーディネータの実行は、各種の通信媒体によって接続できるシステム全体に行き渡るようにしなければならない。現在、このような環境の２相コミット・プロトコルは実現が容易ではない。既存の分散コミット・スコープ・システムは通常、１つの通信マネジャに依存してコーディネータとして機能するか、または分散データベース・マネジャに限定される。また、分散処理システムの各サイトのコーディネータは、システム内の全サイトのＩＤを認識する必要があり、コーディネータが更に複雑になる。

【０００７】分散システムの２相コミット・プロトコルに関しては、より柔軟なアプローチも望まれる。リソース・マネジャは、コーディネータからのＰＲＥＰＡＲＥ信号にある方法でしか応答しない。これらはＣＯＭＭＩＴへの投票を示すＹＥＳか、ＢＡＣＫＯＵＴへの投票を示すＮＯ、または、基本的な２相コミット・プロトコルに対して最適化された場合は、ファイルがＲＥＡＤ　ＯＮＬＹであることを示すＦＯＲＧＥＴである。しかしコーディネータが、通信リソース・マネジャ（ＡＰＰＣ／ＭＶＳなど）に依存しない時は、これらの応答は適切ではないことがある。ＹＥＳ／ＣＯＭＭＩＴ応答により、コーディネータはコミットするリソースが他にない場合でも、作業単位についてログ・レコードの書込みを余儀なくされる。これにより性能は落ち、システム・スループットが下がり、システム再開時間が長くなる。ＮＯ／ＣＯＭＭＩＴ応答では、作業単位がバックアウトされる。ＦＯＲＧＥＴ／ＲＥＡＤ　ＯＮＬＹ応答では、通信リソース・マネジャが、その必要があっても作業単位に関係しなくなる。

6

【０００８】

【発明が解決しようとする課題】本発明の目的は、通信マネジャを使用して、同種または異種オペレーティング・システム間に分散コミット・スコープを提供することである。

【０００９】本発明の目的は、データ処理システムで用いられている通信メカニズムに依存せずに、物理的に離れたエージェント間で分散加入を可能にする２相コミット・プロトコルの拡張機能を提供することである。

【００１０】本発明の目的は、物理的に離れたエージェントが存在する時に、２相コミット・プロトコルの性能を最適化することである。

【００１１】

【課題を解決するための手段】本発明に従って、２相コミット・プロトコルに特別処理ステージが追加される。この特別ステージは第１相終了処理（ＥＰＯＰ）と呼ばれ、任意の通信メカニズムを使用してシステム全体にコーディネータ機能を分散させることができる。第１相終了処理は加入者が制御権を得られる特別ステージである。この特別ステージでは、加入者が２相コミット・プロトコルのシーケンスを分散システムに送る。通信メカニズムは、これが分散コーディネータの一部になるように用いられる。コーディネータ自体は他のシステムの知識を要しない。

【００１２】特別処理ステージは、第１相終了処理イネーブル（ＥＥＰＯＥＰ）と呼ばれるオペレーティング・システムのサービスによって有効になる。特別ステージは、ＥＥＰＯＥＰにより実行側システム上で使用できる。これにより、分散データベースをサポートできるだけでなく、分散したユーザと分散した汎用リソース・マネジャ（障害に耐える保護リソースを所有するプロセスであるリソース・マネジャなど）もサポートされる。

【００１３】本発明の態様に従って、コーディネータからのＰＲＥＰＡＲＥ信号に応答して新しい応答がリソース・マネジャによって使用できる。この新しい応答はＡＢＳＴＡＩＮと呼ばれる。リソース・マネジャからコーディネータへのこの応答は、リソース・マネジャが作業単位の２相コミット・プロセスに関係し続けようとするが、作業単位の最終決定（すなわちＣＯＭＭＩＴまたはＢＡＣＫＯＵＴ）には影響を与えようとしないことを示す。ＡＢＳＴＡＩＮ応答を使用することで、本発明に従った分散された２相コミット・プロトコルの性能が更に最適化される。

【００１４】

【実施例】各図を参照する。図１乃至図４は、仮想端末

7

アクセス方式（ＶＴＡＭ）１２とジョブ入力システム（ＪＥＳ）１３がインストールされたメインフレーム・コンピュータ１１から成るトランザクション処理システムを示す。図に示したＶＴＡＭ１２はアドレス空間であり、そのアドレス空間内に、直接アクセス記憶装置（ＤＡＳＤ）１５と通信するアクセス方式ルーチンがある。ＤＡＳＤにはデータベースとプログラム・ライブラリが格納される。ＶＴＡＭ１２はローカル端末１６とリモート端末１７を伴う。システムのオンライン・ユーザは、複数のローカル端末１６とリモート端末１７を介してデータにアクセスする。ジョブ入力システム（ＪＥＳ）１３は、ローカル・ジョブ入力／出力（Ｉ／Ｏ）装置１８とリモート・ジョブＩ／Ｏ装置１９の両方と通信する。

【００１５】ＶＴＡＭ１２とＪＥＳ１３はいずれも、コンピュータ１１にインストールされる基本オペレーティング・システムと通信する。オペレーティング・システムは、システム・リソースを中央コンソール２２や、機能コンソール２３、及びＪＥＳ１３と共有するマスタ・スケジューラ２１を含む。ＪＥＳ１３は、オペレーティング・システムの一部であるジョブ・スケジューラ２４に入力されるジョブ・キューを生成する。このキューはＶＴＡＭ１２に入力される。

【００１６】オペレーティング・システムは、たとえばＩＢＭの多重仮想記憶（ＭＶＳ）オペレーティング・システムである。ＭＶＳオペレーティング・システムは、タスク・スーパバイザ２５、プログラム・マネジャ２６、タイマ・スーパバイザ２７、仮想記憶マネジャ２８、実記憶マネジャ２９、補助（またはページ）記憶マネジャ３０、復元終了マネジャ３１、入力／出力システム（ＩＯＳ）機能３２などを含む。これらはすべて、各種の割込みハンドラ３４からの入力を受信するシステム状態機能３３と通信し、出力をシステム・リソース・マネジャ３５に提供する。また、タスク・スーパバイザ２５は、ジョブ・スケジューラ２４及びディスパッチャ３６の両方と通信する。ディスパッチャ３６は、中央処理装置（ＣＰＵ）サービスのためにキューイングされたリクエストを持つ。

【００１７】以上は、ＭＶＳオペレーティング・システムの概略である。ＭＶＳオペレーティング・システムの詳細についてはH. Lorin及びH. M. DeitelによるOperatingSystems、Addison-Wesley (1981)、並びにH. M. DeitelによるOperatingSystems、Addison-Wesley (1984)の21章を参照されたい。

【００１８】従来の２相コミット・プロトコル図５は、従来の２相コミット・プロトコルの代表例を示すタイミングチャートである。このタイミングチャートは、２つのリソース・マネジャP_1、P_2を想定している。ユーザはそのいずれかまたは両方を呼出すことができる。リソース・マネジャP_1、P_2は、ＤＢ２、ＩＭＳ／ＤＬ１などのプログラムである。ＤＢ２はＩＢＭデータベース製品で、ＩＭＳ／ＤＬ１はＩＢＭ情報管理システム製品であり、このトランザクション処理システムにＤＬ／１データベースへのアクセスを与えるサービス・プログラムを含む。ただＤＢ２とＩＭＳ／ＤＬ１は一例にすぎず、他のトランザクション処理システムも使用できる。コーディネータは図５ではＣＯＯＲと呼ばれ、図１乃至図４に関して挙げたＩＢＭのＭＶＳ（多重仮想記憶）オペレーティング・システムなど適切なオペレーティング・システムのモジュールでもある。

8

【００１９】ここで図５と図８を参照する。この例のプロセスは図８の機能ブロック１０１から始まり、ここでユーザがデータを変更するかコミット／バックアウトするかを決定する。決定がデータの変更であれば、（この例では）変更されるデータがP_1データかP_2データかがブロック１０２で判定される。P_1データなら、P_1データが機能ブロック１０３で変更される。これは図５ではステップ１として示した。データが変更されるとP_1が独自にこれらデータ変更結果をログする（図８の機能ブロック１０４）。次に図８の機能ブロック１０５と図５のステップ２で、P_1がコーディネータ（ＣＯＯＲ）に、このユーザがそのデータ変更結果を何時コミットまたはバックアウトしようとするかの通知を受けなければならないことを伝える。ここで処理がユーザに戻る。

【００２０】ここでユーザがあるP_2データを変更するとする。その場合、プロセスは判定ブロック１０１、１０２から機能ブロック１０６に進み、そこでP_2データが変更される。図５ではこれをステップ３として示した。P_1のデータ変更と同様にデータが変更された時、P_2は独自にこれらデータ変更結果をログする（図８の機能ブロック１０７）。次に図８の機能ブロック１０８と図５のステップ４で、P_2はコーディネータ（ＣＯＯＲ）に、このユーザが何時そのデータ変更結果をコミットまたはバックアウトするかという通知を受けなければならないことを伝える。この点で処理はユーザに戻る。

【００２１】２相コミット・プロセスは図５のステップ５から始まる。例ではこの時、ユーザが２つの加入者P_1、P_2でそのデータ変更結果をすべてコミットすることに決める。これは、適切なコマンドをコーディネータ（ＣＯＯＲ）に対して直接実行することによって行なわれる（図５のステップ５）。コーディネータは、このコマンドを受信してから、データ変更結果の硬化を２つの加入者P_1、P_2と同期化しなければならない。２相コミット・プロトコルの第１ステップとして、コーディネータは機能ブロック１１０でＰＲＥＰＡＲＥ信号をP_2、P_1に送る（図５のステップ６、７）。コーディネータは次に、P_2、P_1からの投票を待つ（図８の判定ブロック１１１）。

【００２２】図５でP_1はＰＲＥＰＡＲＥ信号にＹＥＳを投じる（ステップ８）。これは、P_1がデータ変更結果を硬化するか、またはバックアウトするかのいずれ

か、コーディネータが指示したとおりに準備が整っていることを示す。図5のステップ9でP₂も同様にPREPARE信号にYESを投じる。すなわちP₂はここで前または後に進む準備が出来ている。この時、コーディネータは加入者から期待していた投票をすべて受信している。コーディネータは、判定ブロック112で投票をもとにして（すべての投票がYESなら）前へ進むか、（すべての投票がYESでなければ）後に進むかを決定する。これはアトミック・インスタントとして知られている。

【0023】この例では、いずれの投票もYESなので、コーディネータは前へ進み、レコードを保証された硬化媒体上のログに書込むことを決定する（機能ブロック113）。図5ではこれをステップ10に"*"で示している。これは決定が行なわれたという意味である。ログに書込みをすることによって、システムが障害を起こすか、またはたとえリソース・マネジャでも障害を起こした時は、コーディネータは再び同期をとり、各加入者に前または後に進むかどうかを伝えることができる。再開時にログにアトミック・インスタント・レコードがある場合、コーディネータは加入者に、ログ・レコードに記録された決定事項に応じて前または後に進むことを伝える。ログにアトミック・インスタント・レコードがない場合、決定が行なわれる前に障害が発生している。その場合の処理はすべてのデータ変更結果をバックアウトすることである。

【0024】図の例の場合、コーディネータは前進する（すなわちコミットする）という信号をP₂に送り（図5のステップ11）、図5のステップ12で後進するという信号をP₁に送る。このコミット機能は図8の機能ブロック114に示した。コーディネータは、前進することを加入者に通知した後、各加入者がそのデータ変更結果を硬化したことを確認応答するまで待たなければならない（図8の機能ブロック115）。図5ではP₁がコーディネータに、そのデータ変更結果の硬化を終えたことを通知し（ステップ13）、P₂はコーディネータに、そのデータ変更結果の硬化を終えたことをステップ14で通知する。これら確認応答が受信されると、コーディネータはユーザP₁とP₂の関係を忘れることができる。従って、コーディネータは別のログ・レコードに書込みをし（図5のステップ15の"#"）、先に書込みをしたアトミック・インスタント・レコードを取消す。これによりコーディネータは、再開時にログを読取る際に作業単位が終了したかどうかを知ることができる。図8の機能ブロック116でログが変更された後、処理はユーザに戻る。

【0025】図5、図8について説明したプロセスは従来の2相コミット・プロトコルの応用である。本発明の背景を説明するためにのみここに記した。加入者P₁、P₂の独自のログ処理の詳細は図5、図8には示していない。ただし、これら"加入者"ログの詳細は分散コミットの流れを理解するうえで大切である。P₁、P₂のログは、図5のステップ10のコーディネータ・ログに示すように、アトミック・インスタント・ログ・レコードを含まない。このレコードを持つのはコーディネータだけである。加入者は代わりに他の一連のエントリ・ログを使用する。これらの加入者のログ・エントリの詳細は、重要ではあるが本発明には含まれない。

【0026】図5のタイミングチャートに示した基本的な2相コミット・プロトコルは、PREPARE信号に別の応答を与えることによって最適化されている。この応答はFORGETまたはREAD ONLYである。図6に、図5のタイミングチャートをもとにし、加入者P₁、P₂の両方がコーディネータからのPREPARE信号に応答してFORGETを投じる場合のタイミングチャートを示す。ステップ10乃至14は省略している。すなわちコーディネータは、FORGET応答をすべて受信した後、前後に進む決定はせず、記録されるアトミック・インスタントはなく、コーディネータはただユーザと加入者P₁、P₂の関係を忘れるだけである（ステップ15）。図7も図5のタイミングチャートをもとにし、P₁がコーディネータからのPREPARE信号にFORGETを投じるが、P₂はYESを投じる場合のタイミングチャートを示す。このケースでは加入者P₁に関係するステップ12、13を略しているが、他のステップはステップ10のアトミック・インスタントを含めてすべて実行される。図9は、図8の流れ図をもとにし、図6、図7に示した例についてREAD ONLY最適化をサポートする変更例を示す流れ図である。図9で判定ブロック112aは図8の判定ブロック112に代わる。すべての投票がYESかどうか判定されるのではなく、少なくとも1つの投票がYESで、NO投票がないという条件がチェックされる。つまり、1つの投票がFORGETである可能性はあるが（図7）、YES応答があるため、機能ブロック113でのログへの書込みに続いて、機能ブロック114aでCOMMIT信号が、FORGETを投じた加入者を除く全加入者に送られる。一方、判定ブロック112aで判定された条件が見つからない場合、更に判定ブロック117で、全投票がFORGETかどうか判定される（このケースは図6）。FORGETの場合、プロセスは直接に機能ブロック116に進み、処理がユーザに戻る前にログが更新される。

【0027】従来の2相コミット・プロトコルは、分散システムにうまく適合しないことが知られている。先に挙げたDateの文献を参照されたい。問題はシステムがどのようなタイプの通信媒体とも接続可能なので（APPC－拡張プログラム間通信、TCP／IP－伝送制御プロトコル／インタネット・プロトコル、OSI／CSなど）、コーディネータの実行結果をシステム全体に行き

渡らさせるのは簡単ではないということである。コーディネータの役割を拡張し、加入者と通信するためのメカニズムから切り離さなければならない。これは本発明に従って、システム間に用いられる通信メカニズムに依存せずにコーディネータの役割を分散させることができる2相コミット・プロトコルの拡張機能によって行なわれる。

【0028】本発明に従った2相コミット・プロトコルの拡張：2相コミット・プロトコルに特別処理ステージが追加される。この特別処理ステージは第1相終了処理（EPOP）と呼ばれ、コーディネータ機能を任意の通信メカニズムを用いるシステム全体に分散させることができる。第1相終了処理は加入者が制御権を得られる特別ステージである。このステージで加入者は、2相コミット・プロトコルのシーケンスを分散システムに送る。通信メカニズムは、どのようなタイプのものでも、それが分散コーディネータの一部になるように用いられる。コーディネータそれ自体は他のシステムの知識を要しない。

【0029】本発明の実施例についての以下の説明に用いる用語は最初に定義しておく必要がある。新しい用語は（本発明に関係する場合は）「」で示す。

【0030】「PREPAREに対するABSTAIN応答」第1相で送られたPREPARE信号にABSTAINを返す加入者は、第1相の終わりに収集された投票に影響を与えない。ABSTAINを返す加入者にはなお、第2相でCOMMITまたはCOMMIT信号が送られ、加入者が第1相終了処理を有効にした場合には第1相終了処理が駆動される。

【0031】「コーディネータのエージェント（AOC）」コーディネータの代理を務めるサブコーディネータ。AOCは、加入者の一部またはユーザの分散の度合いに応じて多くなる。AOCの主な役割は、分散したコーディネータ機能を実行することである。

【0032】「アトミック」アトミックは"ただ1つ"の意味で用いられる。一連のデータ変更がアトミックに行なわれるのは、すべて行なわれるか、または全く行なわれない場合である。ある変更を加えて別の変更を加えない状態は、それらがアトミックに行なわれた時には存在しない。

【0033】「アトミック・インスタント」データ変更結果をすべてコミットするか、またはデータ変更結果をすべてバックアウトするかの決定が成された時の時間点をいう。

【0034】「アトミック・インスタント・レコード（T_{02}ともいう）」コーディネータが決定をコミットまたはバックアウトしたことを示すために硬化媒体に書込まれるレコード。コーディネータしかこのレコードを書込めない。

【0035】「第2相開始ログ・レコード（B_{02}ともいう）」AOCがコーディネータからコミットまたはバックアウトの決定を受信してログしたことを示すために硬化媒体に書込まれるレコード。これは、以下に定義するE_{01}と同様にAOCまたは加入者の活動である。

【0036】「コミット・スコープ」データの変更がすべて1つのシステムで行なわれた場合、コミット・スコープはローカルである。いくつかのデータ変更が2つ以上のシステムで行なわれ、ユーザがこれらの変更をアトミックに行なおうとした場合、コミット・スコープは分散する。

【0037】「コーディネータ」2相コミット・プロセスのコントローラ。分散プロセスでは1つのコーディネータしかない。

【0038】「分散」プロセスは、独立してはいるが関係のあるコンピュータ・システムに存在するいくつかのプロセスから成る場合は、分散している。

【0039】「第1相終了処理イネーブル（EEPOEP）」ローカル・システムに、それがいまコーディネータのエージェント（AOC）であることを通知する（ローカル・コーディネータに通知する）新しいオペレーティング・システム・サービス・コール。すなわちローカル・コーディネータには、それが実際に、分散したコミット・スコープの一部でしかないことが伝えられる。コミット・スコープにはコーディネータが1つしかなく、このサービスにより、ローカル・オペレーティング・システムは、コーディネータが別のシステムに存在し、ローカル・システムがそのコーディネータのエージェントであることを認識する。このサービスはローカル・オペレーティング・システムをコーディネータからAOCに変換する。

【0040】「第1相終了ログ・レコード（E_{01}ともいう）」2相コミットの第1相が完了したことを示すために硬化媒体に書込まれるレコード。これはAOCまたは加入者の活動である。すなわちこのレコードがコーディネータによって書込まれることはない。これは、AOCまたは加入者がコーディネータまたはコーディネータと信じられるエージェントからのPREPARE信号（別のAOCから来ることもある）に応答して投票したことを示す。

【0041】「第1相終了処理（EPOP）」第1相終了レコードがログに書込まれた後に生起する新しい処理ステージ。これは、ローカル加入者がすべて投票を終えたことを示す。更に投票（コミットまたはバックアウト）の結果も示す。これは2相コミット・プロセスを保留し、加入者が投票の結果を分散システムに通信できるようにし、これにより分散コミット・スコープの一部になることができる。

【0042】「ログ」重要データが格納された硬化媒体（通常は磁気ディスク装置）。

【0043】「加入者」2相コミット・プロセスの"コ

ントローリ”（被制御装置）。加入者は、ユーザが使用しているリソース（データベース内のデータや別ユーザへの通信パスなど）を所有する。ユーザは加入者をいくつでも関係させることができる。加入者は分散させることができる。

【００４４】「第１相（２相コミットの）」ＶＯＴＥ（投票）相ともいう。これはＰＲＥＰＡＲＥ信号を加入者に送る相から成る。

【００４５】「第２相（２相コミットの）」ＡＣＴＩＯＮ（実行）相ともいう。これはＣＯＭＭＩＴまたはＢＡＣＫＯＵＴ信号を加入者に送る相から成る。

【００４６】「プロセス」プロセスはコンピュータ・システム内の作業単位であり、ユーザ、プログラム製品、単一スレッド、複数のスレッドなどを含む。

【００４７】「ユーザ」いくつかのシステムに分散しているか分散していないアプリケーション・プログラム。

【００４８】「読取り専用最適化」２相コミット・プロトコルに対する最適化。加入者は第１相で送られたＰＲＥＰＡＲＥ信号にＦＯＲＧＥＴまたはＲＥＡＤ ＯＮＬＹを返す。ＦＯＲＧＥＴを返す加入者には、第２相でＣＯＭＭＩＴまたはＢＡＣＫＯＵＴ信号は送られない。コミット・スコープの全加入者がＰＲＥＰＡＲＥ信号にＦＯＲＧＥＴを返した場合、アトミック・インスタントはログされず、第１相終了（E_{01}）または第２相開始（B_{02}）のレコードを実行する必要がある。

【００４９】図１０にマルチプロセッサ・システムを示す。第１コンピュータ５１（マシンＡ）は通信装置５３を介して第２コンピュータ５２（マシンＢ）に接続される。装置５３は、銅、光ファイバ・ケーブルなど適切な通信リンクの通信媒体を含むことができる。コンピュータ５１、５２はそれぞれ図１乃至図４のコンピュータ・ライクなコンピュータ１１とみることもでき、従って図１乃至図４の詳細はここでは簡単のため省略しているので、図１０のブロック図はハイレベルなブロック図である。

【００５０】図１０に示した２マシン・システムは、本発明に従った分散システムとは何かを理解するうえで役立つ基礎概念を示すが、分散システムを１つのマシン上で実現することも可能である。たとえばＩＢＭの仮想マシン（ＶＭ）オペレーティング・システム、ＶＭ／３７０はＩＢＭシステム３７０コンピュータを管理し、端末から操作する数人のユーザそれぞれが完全なシステム３７０を持っているかのように見せる。また、各ユーザは異なるオペレーティング・システム（ＯＳ）を選択することができる。つまり、ＶＭは一度に複数のオペレーティング・システムを（その仮想マシン上で１つを）実行することができる。ＶＭの詳細については先に挙げたDeitelの文献の第２２章を参照されたい。

【００５１】以下の説明では、マシンＡとマシンＢは物理的に独立したマシン（すなわち実マシン）とみることができ、或いは、仮想マシン・オペレーティング・システム上で走る１つのコンピュータ上の仮想マシンでもよい。もちろん図１０よりも複雑なシステムでは、実マシンと仮想マシンを組合わせて、本発明に従った分散コミット・プロトコルが実現できる分散システムをサポートすることができる。

【００５２】図１０を参照する。コンピュータ５１は、ユーザＡ、通信マネジャ（すなわちＣＭＧＲ＿Ａ）、及びリソース・マネジャＲＭ（すなわち加入者P_1）をサポートするオペレーティング・システム（ＯＳ）５４を実行する。ユーザＡのアドレス空間５５、P_1のアドレス空間５６、及びＣＭＧＲ＿Ａのアドレス空間５７はすべて、ＯＳ５４と通信する。コンピュータ５１には、それぞれＣＭＧＲ＿Ａ、ＯＳ、及びP_1のログが書込まれるＤＡＳＤ５８、５９、６０が接続される。同様にコンピュータ５２は、第２ユーザＢ、通信マネジャ（すなわちＣＭＧＲ＿Ｂ）、及びリソース・マネジャＲＭ（すなわち加入者P_2）をサポートするＯＳ６１を実行する。ユーザＢのアドレス空間６２、P_2のアドレス空間６３、及びＣＭＧＲ＿Ｂのアドレス空間６４はすべてＯＳ６１と通信する。コンピュータ５２には、それぞれＣＭＧＲ＿Ｂ、ＯＳ６１、及びP_2のログが書込まれるＤＡＳＤ６５、６６、６７が接続される。各システム上に示した３つのログは同じ物理記憶装置でも異なる物理記憶装置でもよい。

【００５３】動作時、マシンＡのユーザＡはＣＭＧＲ＿Ａと対話し（破線矢印６８）、それ自体の分散した部分をマシンＢで実行するよう要求する。ＣＭＧＲ＿Ａはオペレーティング・システム５４に、ユーザＡが何時コミットまたはバックアウトするか通知を受ける必要があると通知する。これは破線６９で示した。ＣＭＧＲ＿Ａは次に、通信装置またはリンク５３を介してマシンＢ上のＣＭＧＲ＿Ｂと通信する。ＣＭＧＲ＿ＡとＣＭＧＲ＿Ｂのいずれかまたは両方は、このトランザクションでプレイベート・データをそれぞれＤＡＳＤ５８、６５にログすることができる。次にＣＭＧＲ＿ＢはマシンＢでユーザＢプログラムを起動し（起動されていなかった場合）、オペレーティング・システム６１に、ユーザＢが何時コミットまたはバックアウトしたか通知を受ける必要があることを伝える（破線矢印７０）。マシンＡのユーザＡとマシンＢのユーザＢは、ＡＰＰＣを用いる際には必要に応じて逆にしてもよい。

【００５４】２つのマシンで処理されるトランザクションの一例として、マシンＢのユーザＢがデータベースP_2のローカル・データを変更することを破線矢印７１に示す。その時、P_2はオペレーティング・システム６１に、マシンＡのユーザＡまたはマシンＢのユーザＢが何時コミットまたはバックアウトするかの通知を受ける必要があることを伝える（破線矢印７２）。破線矢印７３は、マシンＡのユーザＡが次にデータベースP_1のロー

カル・データを変更することを示す。破線矢印７３、７１で示したデータベースP_1、P_2の変更は任意の順序で繰返し生じ得る。破線矢印７５はユーザAがすべての変更結果をコミットしたいことを示す。

【００５５】図１１乃至図１３を参照する。図１１はユーザ、コーディネータ（ＣＯＯＲ）、加入者（P_1、P_2）、コーディネータのエージェント（ＡＯＣ）、及び通信マネジャ（ＣＭＧＲ）の役割を、図１０のマシン例について示す。加入者P_1、P_2は、たとえば図５乃至図８の例のようにＤＢ２、ＩＭＳ／ＤＬ１などのデータベースである。ただし本発明はどのような用途にも限定されない。ＣＭＧＲは実際には加入者にすぎないが、コーディネータ及びＡＯＣと協同して分散コミット・プロセスを実現するので区別される。図１２、図１３は、ＥＰＯＰサービスによって有効になった時に行なえる新しいオペレーティング・システム・サービス（ＥＥＰＯＥＰ）と終了処理を示す。

【００５６】図の例でユーザは分散している。プロセスは、データをリソース・マネジャ（加入者P_1など）で変更するマシンA上で実行される。また、データをリソース・マネジャ（加入者P_2など）で変更するマシンB上でもプロセスが実行される。ユーザは次に、すべてのデータ変更結果をコミットするよう要求する。両システムのデータ変更結果はアトミックにコミットされなければならない。ユーザは、その処理を通信マネジャＣＭＧＲ（ＡＰＰＣ、ＴＣＰ／ＩＰ、ＯＳＩ／ＣＳなどの通信マネジャ）で分散することに決める。用いられるメカニズムによって、ユーザはコミット・プロセスをコーディネータからか、または通信マネジャから直接要求する。ＥＰＯＰは、ユーザが何らかの手段によってコミットしたいという希望をコーディネータが知らされる場合は、これに依存しない。

【００５７】上記のとおり、ユーザは図１１乃至図１３の例ではそれ自体をすでに分散している。すなわち関係のある２つのマシン上でプロセスを実行している。ここでも、これらのマシンは実マシン（図１０）か、仮想マシンすなわち仮想マシン・オペレーティング・システムにより１つのコンピュータ上で動作するマシンである。図１１乃至図１３には、処理の流れのうちコミットの部分しか示していない。これが本発明の新規性の元になる部分である。また加入者（P_1、P_2、ＣＭＧＲ＿Ａ、及びＣＭＧＲ＿Ｂ）はすべて、ユーザが何時データ変更結果すべてをコミットまたはバックアウトすることを決定したかの通知を受けたいとコーディネータまたはＡＯＣに伝えている。実現形態によるが、図１１のタイミングチャートに示した時点よりも後に、処理が正常終了したこと（すなわち変更結果がすべてバックアウトされたこと）を示す戻りコードをユーザが受取ることがある。図に示したログ・レコードはすべて、オペレーティング・システムが所有するログにある。加入者は、プライベート・ログを持つこともできるが、これらプライベート・ログは図示していない。

【００５８】図１２のプロセスはマシンAで始まり、判定ブロック２０１で、ユーザが変更結果をコミットしたいかどうか検出される。図１１のステップ１では、ユーザがデータ変更結果をすべてコミットしたことを示す。図１２でコーディネータは、ＰＲＥＰＡＲＥ信号を加入者P_1と通信マネジャＣＭＧＲ＿Ａの両方に同報する（機能ブロック２０２）。これは図１１のステップ２である。図１２でＣＭＧＲ＿Ａは、図１０の通信６８から、分散コミット・シーケンスで別のシステムと通信していることを認識し、機能ブロック２０３（図１１のステップ４）でＰＲＥＰＡＲＥ信号をＣＭＧＲ＿Ｂに送る。これは、用いられている通信マネジャに依存しない。

【００５９】図１３で、プロセスは判定ブロック２０４のマシンBから始まり、ＣＭＧＲ＿Ａからのメッセージが検出される。判定ブロック２０５でＰＲＥＰＡＲＥ信号が検出される。ＣＭＧＲ＿Ｂは、ＰＲＥＰＡＲＥ信号を受取った時、これ（すなわちマシンB）がコーディネータのホームではないことを認識する。これが分かるのはＰＲＥＰＡＲＥ信号を受信したからである。これはコーディネータの”領域”ではないので、コーディネータのエージェント（ＡＯＣ）でなければならない。図１１のステップ５を参照されたい。ＣＭＧＲ＿Ｂはそこで第１相終了処理イネーブル（ＥＥＰＯＥＰ）と呼ばれる新しいオペレーティング・システム・サービスを機能ブロック２０６で実行する。これでオペレーティング・システムに、これがＡＯＣであることが通知される。またこれにより、第１相終了ログ・レコード（E_{01}）が書込まれた後（図１１のステップ１０の後）制御がＣＭＧＲ＿Ｂに渡る。次に処理はＣＭＧＲ＿Ｂに戻る。

【００６０】ここでＡＯＣは、ＰＲＥＰＡＲＥ信号をP_2とＣＭＧＲ＿Ｂに同報する（機能ブロック２０７）。これは、（１）ユーザがコミット・リクエストを実行することを通信マネジャによって指示されたか、または（２）ＡＯＣが、新しいＥＥＰＯＥＰサービス・コールを受信した時がコミット・プロセスを開始する時と仮定したことのいずれかに応じて行なわれる。図１１のステップ６を参照されたい。図１１に示した例の場合、ＣＭＧＲ＿Ｂはステップ７でＰＲＥＰＡＲＥ信号にＹＥＳを投じ、P_2は、ステップ８でＰＲＥＰＡＲＥ信号にＹＥＳを投じる。これらの投票は図１３では判定ブロック２０８で検出される。

【００６１】次にＡＯＣは、（１）その投票をすべての加入者（ＣＭＧＲ＿ＢとP_2）から受信し、（２）図１１のステップ５で実行されたＥＥＰＯＥＰサービスによって、それがＡＯＣであることがこのオペレーティング・システム・インスタンスに通知されたので、機能ブロック２０９で第１相終了（E_{01}）レコードをログに書込

む。このレコードを書込むのはAOCだけである。このレコードは図11のステップ9のタイミングチャートに"*"で示した。次にAOCは、CMGR_Bが先に第1相終了処理イネーブル（EEPOEP）サービスを実行していたので、ステップ10で制御をCMGR_Bに渡す。これは、第1相終了と呼ばれる新しい処理ステージである。CMGR_Bは、PREPARE信号の結果を通知される。第1相終了では、CMGR_Bが次にCMGR_AにYESを返す（図13の機能ブロック210）。図11のステップ11を参照されたい。これは、ステップ4でCMGR_AがCMGR_Bに送ったPREPAREへの応答である。図の例では、すべての投票がYESであり、PREPAREの結果はYESだった（すなわちコミット処理に進む）。PREPARE信号は他の結果もあり得るが、本発明の説明を補足するものではないので省略する。

【0062】図11のステップ3でP_1は、それが準備を選択することを示し（この例ではYES）、これは、それが行なったデータ変更の結果をすべてコミットまたはバックアウトする準備が出来ていることを示す。P_1による投票とCMGR_Bによって返った投票は、図12の判定ブロック211で検出される。CMGR_Aはまた、ステップ2でそれにローカルに送られたPREPARE信号にYESを投じる。これは、全加入者（他のマシンに分散することもある）がいまPREPARED STATEにあることを認識して行なう。この知識はステップ11から得られる。すなわち全加入者が前か後に進む準備が出来ている。これは図11のステップ12で生じる。

【0063】マシンAのコーディネータはここで、それが予測していたすべての投票を受取っており、この例では全投票がYESである（判定ブロック212）。そこでコーディネータは、前進してアトミック・インスタント・レコード（T_{02}）をログに書込むことを決定する（機能ブロック213）。このT_{02}レコードは、図11のステップ13に"*"で示した。コーディネータはステップ14でコミット・オーダをローカル加入者P_1とCMGR_Aに同報する。これは図12では機能ブロック214に示した。CMGR_Aは、コミット・オーダを受信した時、コミット・オーダをCMGR_Bに送る（機能ブロック215）。これは図11のステップ15である。

【0064】図13で、CMGR_Aからのメッセージが再び判定ブロック204で検出される。ただこの時のメッセージはCOMMITである（判定ブロック205）。CMGR_Bは、決定（例ではYESなど）をAOCに返すことによってその第1相終了処理を終えている。図11ではこれはステップ17である。ここでAOCは第1相終了処理が完了し、コーディネータ領域からの決定は前進（すなわちコミット）であることを認識する。AOCは第2相開始（B_{02}）レコードをローカル・ログに書込む（図13の機能ブロック216）。復元シナリオのこの時点から以降、ローカル・システムはコーディネータとの通信を再確立してコミット／バックアウト決定を得る必要がない。決定はB_{02}ログ・レコードに記録される。このレコードが書込まれるのは、ローカル・システムがEEPOEPサービスによってAOCになったからである。B_{02}レコードは、コーディネータによって書込まれることはなく、AOCだけが書込む。このB_{02}レコードは図11では"*"で示した。

【0065】AOCは、コミット・オーダをマシンBのローカル加入者P_2とCMGR_Bに同報する（図13の機能ブロック217）。図11ではこれをステップ19に示した。AOCは、ログ・レコードの書込みと第1相終了処理（EPOP）以外はコーディネータと同じように機能する。CMGR_Bは機能ブロック218でCMGR_Aに、データ変更結果のコミットを終えたこと（すなわちB_{02}レコードがログ上で安全であること）を通知する（図11のステップ20）。CMGR_Bは図11のステップ22で、ステップ29のコミット同報からAOCに戻る。これによりAOCは、CMGR_Bがそのコミット処理を終えたとの通知を受ける。またP_2は、ある点で（図11のステップ23など）ステップ19のコミット同報からAOCに戻る。これは図13の判定ブロック219で検出され、CMGR_BとP_2がデータ変更結果のコミットを終えたことがAOCに通知される。AOCは、全加入者がそのデータ変更を終えたことを認識した後はいつでも（図11のステップ25など）、これを機能ブロック220でログに記録する。これは性能の改良であり、再開処理を高速化する。マシンBのAOCは、それが分散コミット・スコープに加わったことを認識していたが、用いられた通信メカニズムまたはプロトコルは認識しなかった。書込まれたレコードは図11のステップ25では"X"で示した。

【0066】P_1は、ある点で（図11のステップ16）、それがデータ変更結果のコミットを終えたことをマシンAのコーディネータに通知する。図11のステップ21でCMGR_AもステップI4のコミット同報からコーディネータに戻る。これは、図12の判定ブロック221で検出される。全加入者がそのデータ変更を終えたことをコーディネータが認識した後はいつでも（図11のステップ24など）、コーディネータはこれをログに記録する（機能ブロック222）。これは性能の改良であり再開処理を高速化する。

【0067】コーディネータは、それが分散コミット・スコープに加わったことを認識していなかった。このレコードは図11のステップ24には"X"で示した。図11の例のキーポイントは、処理シーケンスを示し、データが常に調整されるようにすることである。すなわちユーザのデータの一部だけを更新することは常に許されな

い。変更はすべて要求に応じて行なわれるか、または、何か問題がある場合はすべての変更が逆にされる。この新しいステップ（EPOPとEEPOEP）はデータの保全性を保つ。図１１の例では、前進する判定が成された。バックアウト判定も行なえる。これもログに記録され同様の処理が続く。

【００６８】図１１に示した例の処理により、障害が発生した場合にはいつでもすべてのデータが調整されるようになる。これは、データの一貫性が保たれると表現されることもある。

【００６９】ここで説明を簡単にするために、障害は図１１に示した分散システムの両方が共に障害を起こしたものとする。これより簡単なケースで、システムが１つだけ障害を起こした場合は、両方が同時に障害を起こした最悪のシナリオに含まれる。同様に、両方が故障したが同時にではない場合も、用いられているログ・レコードが特定の時間順で書込まれるという事実により扱える。また、分散コミット・スコープに参加している分散システムが２つ以上の場合も同様である。これと同じロジックは、高さが３以上、幅が２以上のツリーに展開される。図１１の例では、ツリーの高さは２で、親であるマシンAの子は１つ、幅は１になる。

【００７０】下記の障害例では図１１を参照する。

【００７１】例１：E_{01}ログ・レコードがログに書込まれる前に障害が発生する（すなわちAとした時間より前）。再開時に、マシンAで、P_1がそのプライベート・ログ（図示なし）を読取り、ユーザのためにデータを変更していたことを発見する。P_1は、オペレーティング・システムにコミットまたはバックアウトの決定を依頼する。オペレーティング・システム（すなわちコーディネータ）は、"ユーザ"に関係するそのログでT_{02}またはB_{02}を発見せず、これをP_1に通知する。P_1はそこですべてのデータ変更結果をバックアウトする。CMGR__Aはそのプライベート・ログを読取り、それがユーザのためにデータを変更していたことを発見する。CMGR__Aはオペレーティング・システムにコミットまたはバックアウトの決定を依頼する。オペレーティング・システム（すなわちコーディネータ）は、ユーザに関するそのログでT_{02}またはB_{02}を見つけず、これをCMGR__Aに通知する。CMGR__Aは次に、ある場合はすべてのデータ変更結果をバックアウトするか、またはこのユーザを無視する。マシンBで、P_2はマシンAの場合と同様、このステップの処理を同じ実行結果で行なう。CMGR__Aは、マシンAの場合と同様、このステップの処理を同じ実行結果で行なう。

【００７２】例２：T_{02}ログ・レコードが時間Bに書込まれる前の時間AにE_{01}ログ・レコードが書込まれた後に障害が発生する。再開時、マシンAで、P_1は例１のように処理を実行する。CMGR__Aは例１のように処理を実行する。マシンBで、P_2は、例１のように処理を実行し、オペレーティング・システムによって、ユーザの状態がIN_DOUBTであると通知される。P_2はこれを、コーディネータが（別のマシンで）T_{02}レコードを書込んだかどうかをローカル・システムがまだ認識していないと解釈する。すなわちコーディネータが決定をコミットまたはバックアウトしたかどうか認識しない。その後P_2は通知を受ける。すなわちコーディネータとの再同期化が完了した後である。P_2は、それがまだすべての変更をコミットまたはバックアウトする準備を整えた状態のままでなければならないことを覚えておく必要がある。CMGR__Aは例１のように処理を実行し、オペレーティング・システムによって、ユーザの状態がIN_DOUBTであると通知される。CMGR__Aは、それがログに格納した情報から、コーディネータ（またはおそらくは単に他のAOC）がマシンAにあることを認識する。そのログから、CMGR__Aとの通信を再確立するのに充分な情報を読取る。次にCMGR__Aに、コミットまたはバックアウトの判定を求める。CMGR__Aは、それが再開操作を終えている場合は、T_{02}またはB_{02}ログ・レコードが見つからなかったことを認識する。CMGR__AはCMGR__Bに、判定がバックアウトであることを通知する。CMGR__BはAOCにこれを通知し、AOCは、対象加入者に判定がバックアウトであることを通知する。例でAOCはP_2に通知する。

【００７３】例３：B_{02}ログ・レコードが時間Cに書込まれる前に（時間Bなどに）T_{02}ログ・レコードが書込まれた後で障害が発生する。再開時、マシンAで、P_1は例１のように処理を実行し、ユーザが状態IN_COMMITにあることをオペレーティング・システムによって通知される。P_1はこれを、ユーザのためのデータ変更をコミットしなければならないと解釈し、そう実行する。CMGR__Aは、例１のように処理を実行し、ユーザが状態IN_COMMITにあるという通知を受ける。CMGR__Aはこの情報を、（１）CMGR__AがCMGR__Bと再同期化するか、または（２）CMGR__BがCMGR__Aにユーザの判定を求めた時に、CMGR__Aに引渡す。CMGR__AはこれをAOCに通知し、AOCはそこで、すべての対象加入者に判定がコミットであることを通知する。例でAOCはP_2に通知する。マシンBで、P_2は、例２のように処理を実行する。CMGR__Bは例２のように処理を実行するが、T_{02}レコードが見つかったことをCMGR__AがCMGR__Bに通知する。従って、判定はすべての変更をコミットすることである。CMGR__BはこれをAOCに通知し、AOCはそこですべての対象加入者（例ではP_2）に通知する。

【００７４】例４：最終レコードのいずれか（図１１のステップ２４、２５）が書込まれる前にB_{02}レコードが（時間Cで）書込まれた後に障害が発生する。その場合

P_1、またはP_2は例1のように処理を実行し、ユーザの状態がIN_COMMITであることをオペレーティング・システムから通知される。従ってP_1、P_2はすべてのデータ変更結果をコミットする。一方、CMGR_AとCMGR_Bは例1のように処理を実行し、ユーザの状態がIN_COMMITであることをオペレーティング・システムから通知される。用いられているプロトコルによるが、CMGR_AとCMGR_Bは、それらが再同期化した時にこの情報を交換する。ただし、コミットまたはバックアウトの判定は、各システムで使用できるログ・レコードから個別に推定できるので、再同期化時に交換された情報は用いられない。

【0075】例5：最終レコードのいずれかまたは両方が時間D、Eで書込まれた後に障害が発生する。その場合P_1、P_2は、（1）ユーザの状態がIN_COMMITであることをオペレーティング・システムから通知され、すべてのデータ変更結果をコミットするか、または（2）ユーザの代わりに行なわれたすべての作業がすでに終わっており、何の実行も必要ないことが通知される（または推定することができる）。一方、CMGR_AとCMGR_Bは、（1）ユーザの状態がIN_COMMITであることをオペレーティング・システムから通知され、例4のように処理を実行するか、または（2）ユーザの代わりに行なわれたすべての作業がすでに終わっており、何の実行も必要ないことが通知される（または推定することができる）。

【0076】以上、復元フローについて説明したが、これは加入者がここに示したプロシジャに従う際、データの保全性が常に保たれることを示すものである。それ以上のデータ保全性は、標準的な2相コミットについては明らかにされていない。EEPOEPサービスと第1相終了処理により、分散コミット・スコープが有効になり、分散コーディネータが必要でなくなる。すなわちコーディネータは、特定の通信プロトコルまたは分散ネットワーク・トポロジを認識もせず、それに対応もしなかったということである。EEPOEPサービスでコーディネータのエージェント（AOC）領域を形成することによって、システムの数にかかわらずにコミット・スコープの分散を維持できる。例では2つのシステムしかコミット・スコープに参加していないが、この機構を用いることで、参加できるシステムの数は制限されない。

【0077】コーディネータの機能はデフォルトであるが、AOCの機能は、新しいオペレーティング・システム・コール（すなわちEEPOEP）1つで実現できる。それでもシステム間通信の流れは標準的な2相コミットである。すなわち例では、マシンAとマシンB間の流れは拡張機能がなく2相コミットとみられる。コーディネータ・システムはなお標準的な2相コミットとみられる。すなわち例では、マシンAの流れは標準的とみられる。マシンAのコーディネータは、コミット・スコープが分散していることを認識さえしない。適切な時間にEEPOEPサービスによって、マシンBがコーディネータからコーディネータのエージェント（AOC）に変換される。

【0078】上記の内容から明らかなように、本発明は、コーディネータを効率よく分散させるが、コーディネータはこの事実の知識をもっている必要はなく、大幅に簡素化される。コーディネータ機能は、任意の通信メカニズムを使用して分散される。ユーザは分散コーディネータ機能を意識する必要がない。更に、汎用リソース・マネジャは、それに合った通信マネジャを使い、この新しいプロトコル拡張機能を認識し、オペレーティング・システム同期点マネジャ（すなわちコーディネータ）と通信することによって分散させることができる。この方法の新規性は、コーディネータ機能が通信媒体に依存せずに分散されることである。コーディネータにより、特定の通信製品の知識がないまま分散コミット・スコープ（すなわち拡張2相コミット）が有効になる。第1相終了処理という拡張機能では、この分散を、分散しているコーディネータ機能をオペレーティング・システムが認識しないままに実現することができる。

【0079】このほか、分散加入者に対する2相コミット・プロトコルの拡張機能として、コーディネータからのPREPARE信号に対してリソース・マネジャが新しい応答を使用することができる。この応答は、PREPARE信号に応答してコーディネータのエージェントに応答するために通信リソース・マネジャ（CMGR_A）が使用できるABSTAIN応答である。この新しい応答により、性能向上、システム・スループットの向上、及び分散システム上の再開時間の短縮が可能になる。

【0080】図14に、全加入者がマシンBでFORGETまたはABSTAINを、マシンAでFORGETを選択する場合について、分散2相コミットのタイミングチャートを示す。図14は図11をもとにしており、ステップはいくつか削除している。

【0081】従来の2相コミット・プロトコルでは、リソース・マネジャがコーディネータからのPREPAREリクエストにある方法で応答する必要がある。加入者はそのリクエストに対して、コミット・コーディネータへのYES／COMMIT、NO／BACKOUT、またはFORGET／READ ONLYで応答しなければならない。しかし、コーディネータのエージェントが、YES／COMMIT、NO／BACKOUT、またはFORGET／READ ONLYの応答を可能にするだけであれば、読取り専用最適化を行なうことはできない。これは、CMGR_Bが図11のステップ7でPREPAREにYESと応答しなければならないという事実による。これによりAOC_Bは、更新されたコミット可能なリソースがマシンBにあるとみなし、従っ

て、加入者P_2がPREPARE信号にFORGET／READ ONLYを返した場合でも、2相コミット・プロセスを完全に実行しなければならない。図14のステップ7では、CMGR_BがAOC_BからのPREPARE信号にABSTAINを選択する。CMGR_BからのこのABSTAIN投票により、図10の次のステップは生じない。

1．ステップ9でAOC_Bによって第1相終了（E_{01}）がログに書込まれる。

2．ステップ11で通信応答YESがマシンAに送られる。

3．ステップ15で通信応答COMMITオーダがCMGR_Bに送られる。

4．ステップ18でマシンBのAOC_Bによって第2相開始（B_{02}）が書込まれる。

5．ステップ19、22、23のCOMMIT信号と応答。

これらのイベントはすべて、マシンBが実際に、図11の加入者P_2のように、2相コミット・プロトコル・リソースを持っている時に必要である。

【0082】加入者P_2が作業単位に関係しないか、加入者P_2しか読取り専用リソースを持っていない場合、これらのステップすべてが必要なわけではない。こうした状況で性能を向上させる2相コミット・プロセスのREAD ONLY最適化がある。図14のタイミングチャートはREAD ONLY最適化を示す。不要なプロセスは上記のログへの書込みと通信イベントなどである。CMGR_Bはステップ7でPREPARE信号にNOを返すことができない。これにより作業単位がパックアウトされる。CMGR_Bは、CMGR_Bが作業単位に関わろうとしなくなったとAOC_Bがみなすことになるので、FORGETを返すことができない。

【0083】ABSTAINは、本発明によって提供される新しい応答であり、CMGR_BはこれをPREPARE信号への応答に使用することができる。この応答は、ABSTAINで応答する加入者（例ではCMGR_B）が、PREPAREリクエストの最終結果に影響を与えようとしないこと、どのような結果（YES、NO、またはFORGET）も受入れようとすること、及び、現在の作業単位の2相コミットに関係し続けようとすること、を示す。図14のステップ7では、CMGR_Bは、PREPARE信号にYESと答えるのではなく、ABSTAINを返す。加入者P_2がPREPARE信号にYESと答えた場合、AOC_BのPREPARE信号の最終結果はYESであり、イベントは上記のように進む。加入者P_2がFORGETと答えるか、加入者P_2が存在しない場合、AOC_B PREPARE信号の最終結果はFORGETである。AOC_Bは、図14のステップ10でFORGETの結果をCMGR_Bの第1相終了処理に返す。CMGR_Bはここで、ステップ11でのYES投票ではなく、通信応答FORGETをマシンA送る（ステップ20）。これによりマシンAは、作業単位についてマシンBと接触しなくなる。CMGR_Bは次にAOC_Bに戻り、AOC_Bはステップ25で"単位復元終了"レコードをログに書込み、作業単位についてのその処理を終える。

【0084】図14のタイミングチャートを図11と比較すると、マシンBに読取り専用リソースがある場合、マシンBの以下のステップはなくなる。

9）第1相終了（E_{01}）をログする。

11）PREPAREに対してマシンAにYES応答を返す。

15）マシンAからコミット・オーダを通信する。

18）第1相開始（B_{02}）をログする。

19）コミット・オーダをマシンBの加入者に引渡す。

22）CMGR_Bのコミット終了をAOC_Bに返す。

ステップ20はステップ10の後、ステップ12の前に移動する。以下のマシンA上のイベントは、マシンBが読取り専用リソースを持っている時にはなくなる。

14）コーディネータからCMGR_Aへコミット終了コールを出す。

21）CMGR_Aのコミット終了から復帰する。

【0085】図16は、図12の流れ図にもとづく流れ図で、図14のタイミングチャートに示したマシンAでの操作の変更を示す。判定ブロック212aで、YES投票が1つ以上でNO投票がないという条件がチェックされる。これはもちろん図9の判定ブロック112aで行なわれるチェックと同様である。また機能ブロック215はなくなり、機能ブロック214aは変更され、COMMITが加入者P_1にしか送られない。これは例でCMGR_AがFORGETを選択したからである。

【0086】図17は、図13の流れ図にもとづく流れ図で、図14のタイミングチャートに示したマシンBにおける操作の変更を示す。この流れ図で判定ブロック205と機能ブロック209、210、216、217はなくなっている。これらのブロックの代わりに新しい判定ブロック224がある。ここで受信された投票がすべてFORGETかABSTAINかが判定される。その場合、プロセスは機能ブロック218に進み、ACK信号がCMGR_Aに送られる。他の場合は処理がユーザに戻る。

【0087】マシンA、Bが、コミット対象の更新されたリソース（すなわち読取り専用）を持たない場合、マシンAから削除できるイベントがある。

【0088】13）"アトミック・インスタント"コミット・レコードT_{02}の強制ログ書込み、これにより性能が更に向上する。図15のタイミングチャートにこのイベントを示す。図18は、図12の流れ図にもとづく流れ図で、図15のタイミングチャートに示すマシンAにお

ける操作の変更を示す。判定ブロック２１２ｂは、すべての投票がＦＯＲＧＥＴかＡＢＳＴＡＩＮかをチェックし、機能ブロック２１３、２１４、２１５、及び判定ブロック２２１はなくなっている。マシンＢにおける操作は、図１５に示したタイミングチャートについての図１７の流れ図と同じである。

【００８９】本発明は、分散データベースに、或いはマルチプロセッサ・システムにも限定されない。どのようなリソース・マネジャでも、或いはユーザでも、１つのマシン或いは任意個数のマシン上で分散させることができ、１つの大きなコミット・スコープに参加することができる。

【図面の簡単な説明】

【図１】本発明を実現できるトランザクション処理システムのブロック図である。

【図２】本発明を実現できるトランザクション処理システムのブロック図である。

【図３】本発明を実現できるトランザクション処理システムのブロック図である。

【図４】本発明を実現できるトランザクション処理システムのブロック図である。

【図５】現在の２相コミット・プロトコルの基本操作を示すタイミングチャートである。

【図６】２相コミット・プロトコルのＲＥＡＤ　ＯＮＬＹ最適化の操作を示すタイミングチャートである。

【図７】２相コミット・プロトコルのＲＥＡＤ　ＯＮＬＹ最適化の操作を示すタイミングチャートである。

【図８】図５に示した現在の基本的な２相コミット・プロトコルの流れ図である。

【図９】図６、図７に示したＲＥＡＤ　ＯＮＬＹ最適化を実現する２相コミット・プロトコル・プロセスの流れ図である。

【図１０】本発明に従った拡張２相コミット・プロトコルの操作を説明するのに役立つマルチプロセッサ・システムのブロック図である。

【図１１】本発明に従った、図１０の２マシン環境における２相コミット・プロトコルの拡張機能の操作を示すタイミングチャートである。

【図１２】２相コミット・プロトコル・プロセスの拡張機能について、マシンＡにおけるプロセスを示す流れ図である。

【図１３】２相コミット・プロトコル・プロセスの拡張機能について、マシンＢにおけるプロセスを示す流れ図である。

【図１４】マシンＢにおいて投票がすべてＦＯＲＧＥＴまたはＡＢＳＴＡＩＮであり、マシンＡにおいてＣＭＧＲ＿ＡがＦＯＲＧＥＴを選択した場合について、ＲＥＡＤＯＮＬＹ最適化と新しいＡＢＳＴＡＩＮ応答をサポートする２相コミット・プロトコルの拡張操作を示すタイミングチャートである。

【図１５】マシンＡ、Ｂの両方において投票がすべてＦＯＲＧＥＴまたはＡＢＳＴＡＩＮである場合について、ＲＥＡＤ　ＯＮＬＹ最適化と新しいＡＢＳＴＡＩＮ応答をサポートする２相コミット・プロトコルの拡張操作を示すタイミングチャートである。

【図１６】図１４に示した例について、２相コミット・プロトコル・プロセスの拡張機能を改良するマシンＡにおけるプロセスの流れ図である。

【図１７】図１４に示した例について、２相コミット・プロトコル・プロセスの拡張機能を改良するマシンＢにおけるプロセスの流れ図である。

【図１８】図１５に示した例について、２相コミット・プロトコル・プロセスの拡張機能を改良するマシンＡにおけるプロセスの流れ図である。

【符号の説明】

１１　メインフレーム・コンピュータ
１２　仮想端末アクセス方式（ＶＴＡＭ）
１３　ジョブ入力システム（ＴＥＳ）
１５　直接アクセス記憶装置（ＤＡＳＤ）
１６　ローカル端末
１７　リモート端末
１８　ローカル・ジョブ入力／出力（Ｉ／Ｏ）装置
１９　リモート・ジョブＩ／Ｏ装置
２１　マスタ・スケジューラ
２２　中央コンソール
２３　機能コンソール
２４　ジョブ・スケジューラ
２５　タスク・スーパバイザ
２６　プログラム・マネジャ
２７　タイマ・スーパバイザ
２８　仮想記憶マネジャ
２９　実記憶マネジャ
３０　補助（またはページ）記憶マネジャ
３１　復元終了マネジャ
３４　割込みハンドラ
３５　システム・リソース・マネジャ
３６　ディスパッチャ
５１　第１コンピュータ
５２　第２コンピュータ
５３　通信装置
５４　オペレーティング・システム（ＯＳ）
５５　ユーザＡのアドレス空間
５６　P_1 のアドレス空間
５７　ＣＭＧＲ＿Ａのアドレス空間
６２　ユーザＢのアドレス空間
６３　P_2 のアドレス空間

【図1】

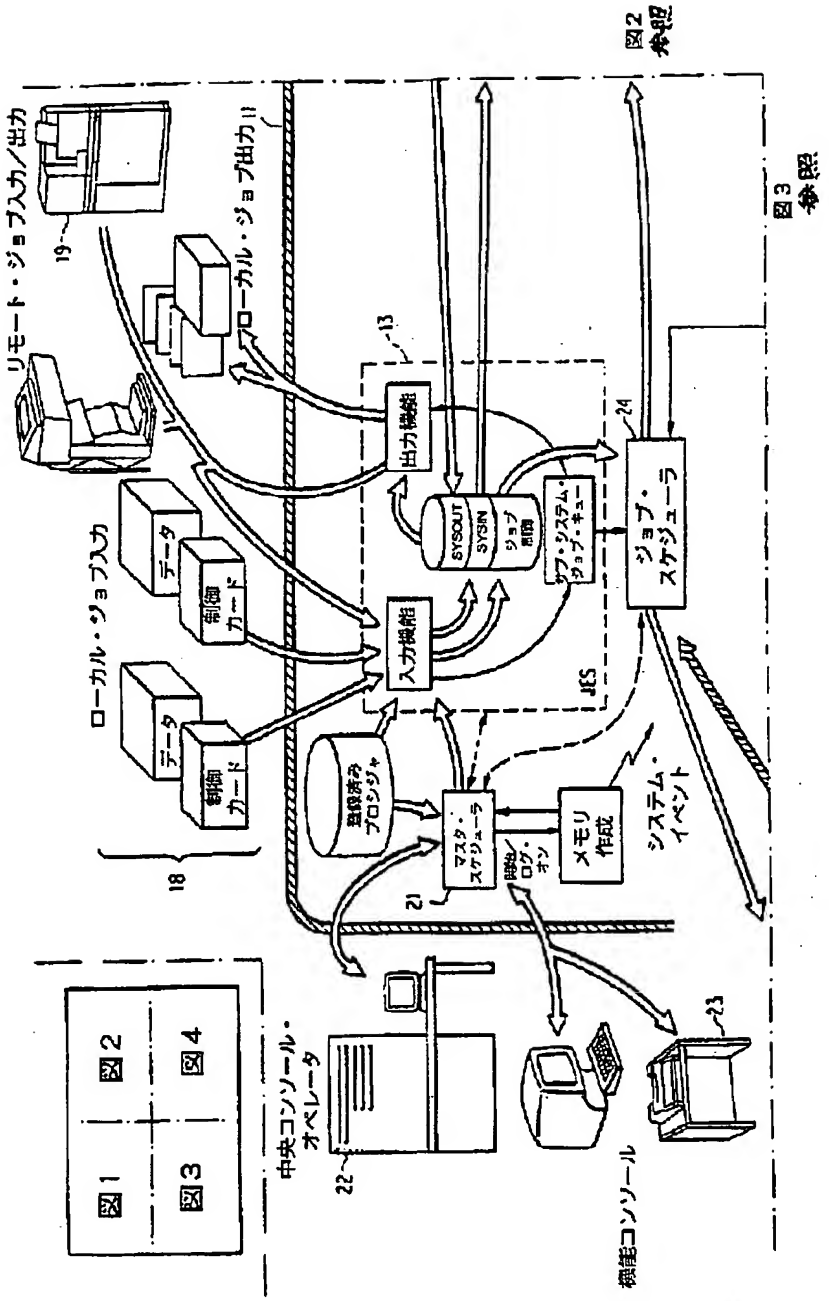


図2 参照
図3 参照
リモート・ジョブ入力／出力
19
ローカル・ジョブ出力
11
13
出力機能
SYSOUT
SYSIN
ジョブ制御
サブ・システム・ジョブ・キュー
24
ジョブ・スケジューラ
ローカル・ジョブ入力
データ
制御カード
データ
制御カード
18
入力機能
JES
登録済みプロシジャ
マスタ・スケジューラ
21
開始ログ・オン
メモリ作成
システム・イベント
図1
図2
図3
図4
中央コンソール・オペレータ
22
23
機能コンソール

【図2】

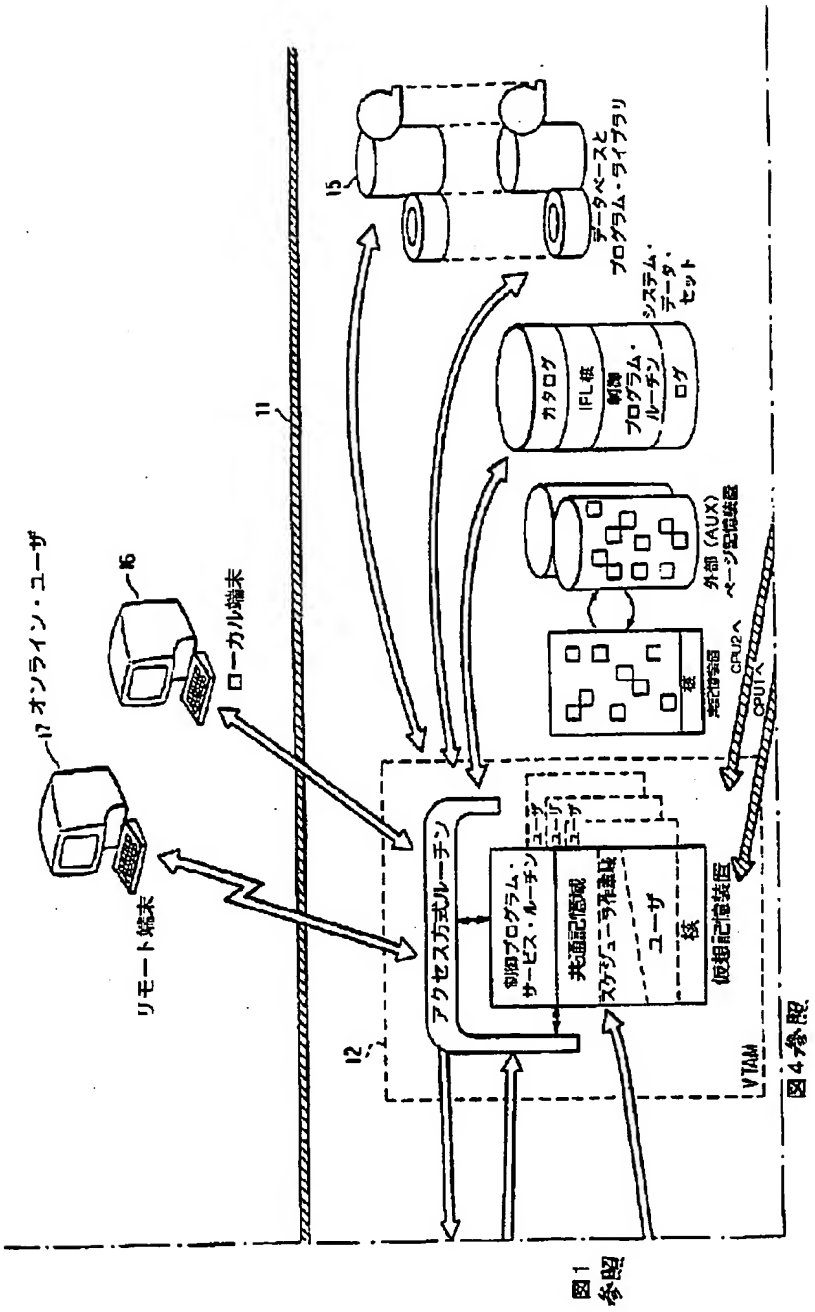

データベースと
プログラム・ライブラリ
システム・
データ・
セット
カタログ
IFL様
制御
プログラム・
ルーチン
ログ
外部（AUX）
ページ記憶装置
CPU2へ
CPU1へ
オンライン・ユーザ
ローカル端末
リモート端末
アクセス方式ルーチン
制御プログラム・
サービス・ルーチン
共通記憶域
スケジューラ作業域
ユーザ
様
仮想記憶装置
VTAM
15
16
17
11
12

図1参照

図4参照

【図3】

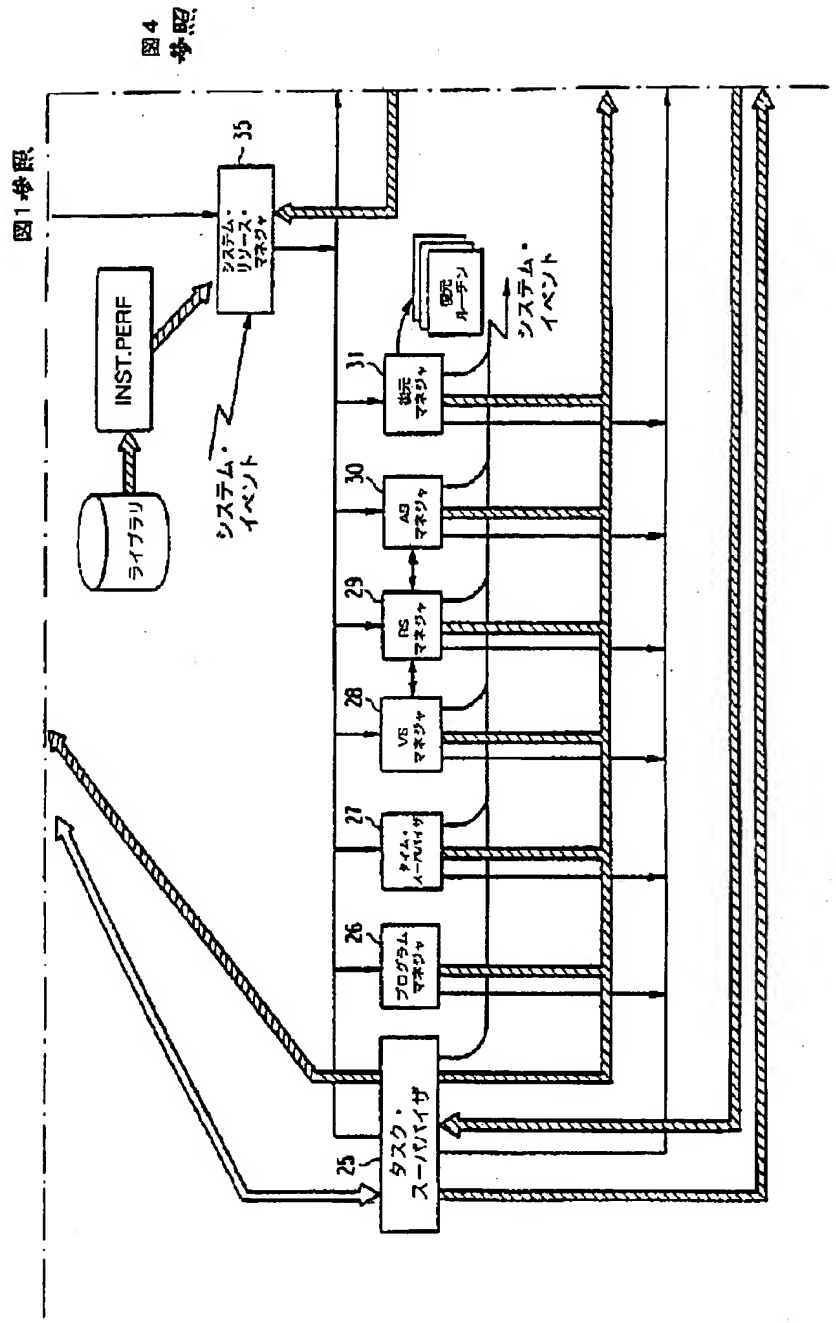


図4参照
図1参照
35
システム・リソース・マネジャ
INST.PERF
ライブラリ
システム・イベント
システム・イベント
31
30
29
28
27
26
25
タスク・スーパバイザ

【図4】

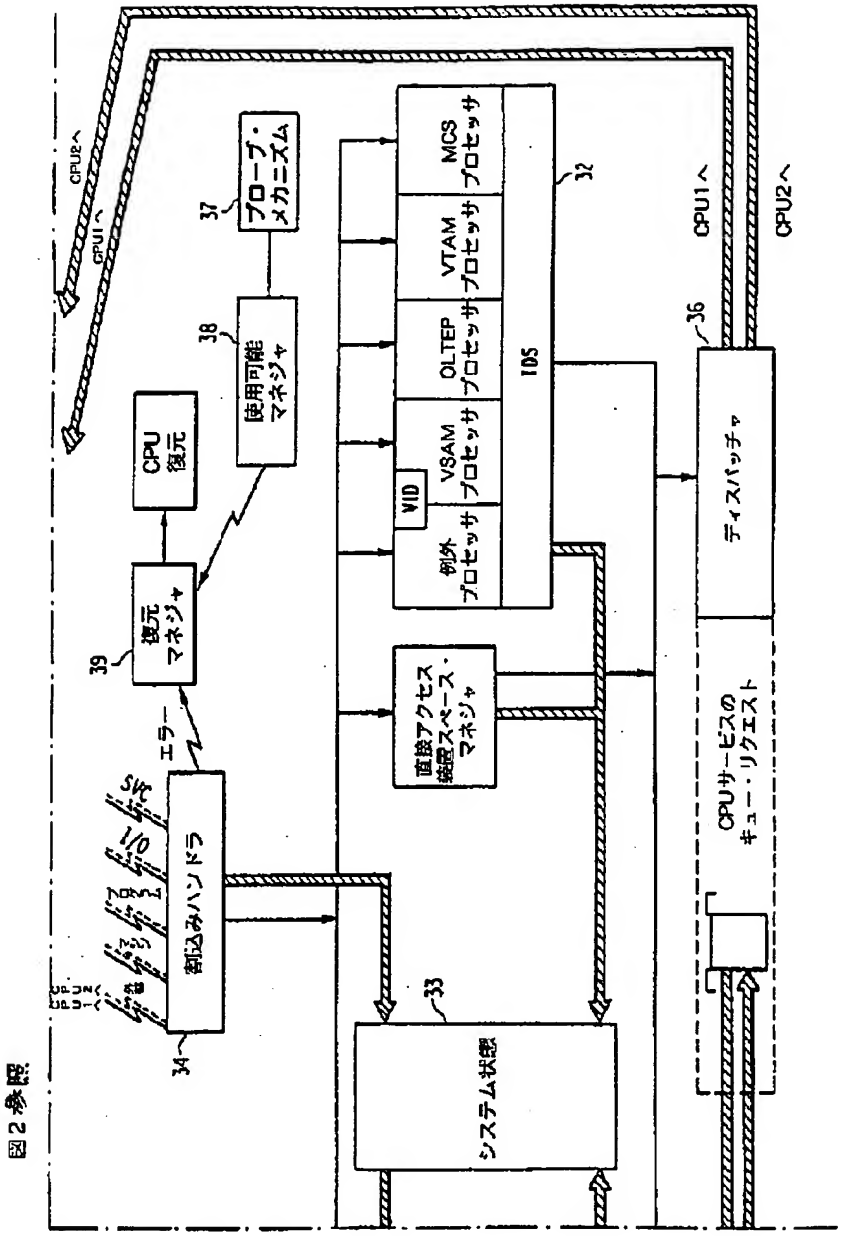
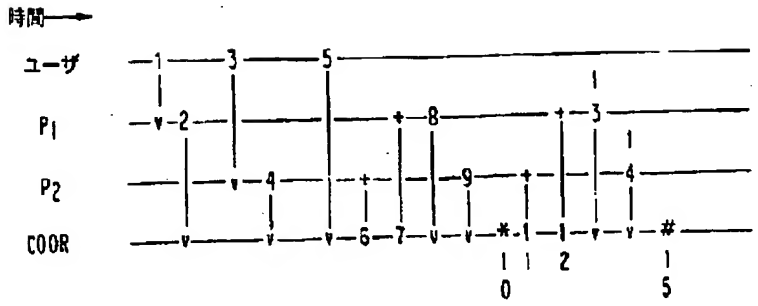

CPU2へ
CPU1へ
37
プローブ・メカニズム
38
使用可能マネジャ
CPU復元
39
復元マネジャ
エラー
割込みハンドラ
34
33
システム状態
直接アクセス装置スペース・マネジャ
例外プロセッサ
VID
VSAM プロセッサ
OLTEP プロセッサ
VTAM プロセッサ
MCS プロセッサ
32
IOS
36
ディスパッチャ
CPUサービスのキュー・リクエスト
CPU1へ
CPU2へ

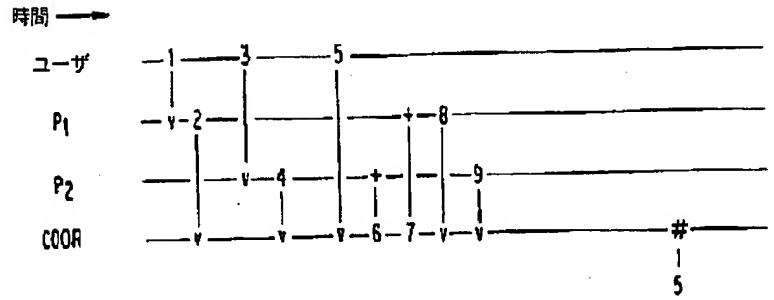
図2参照

図3参照

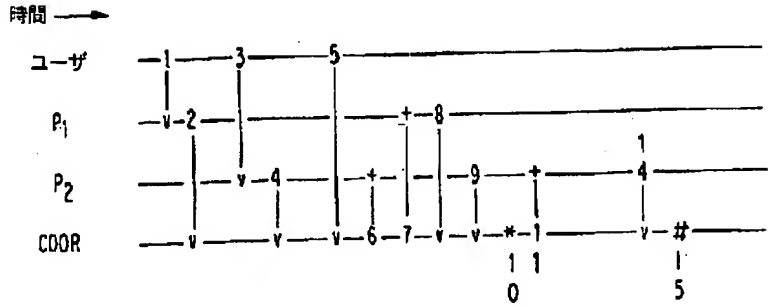
【図5】



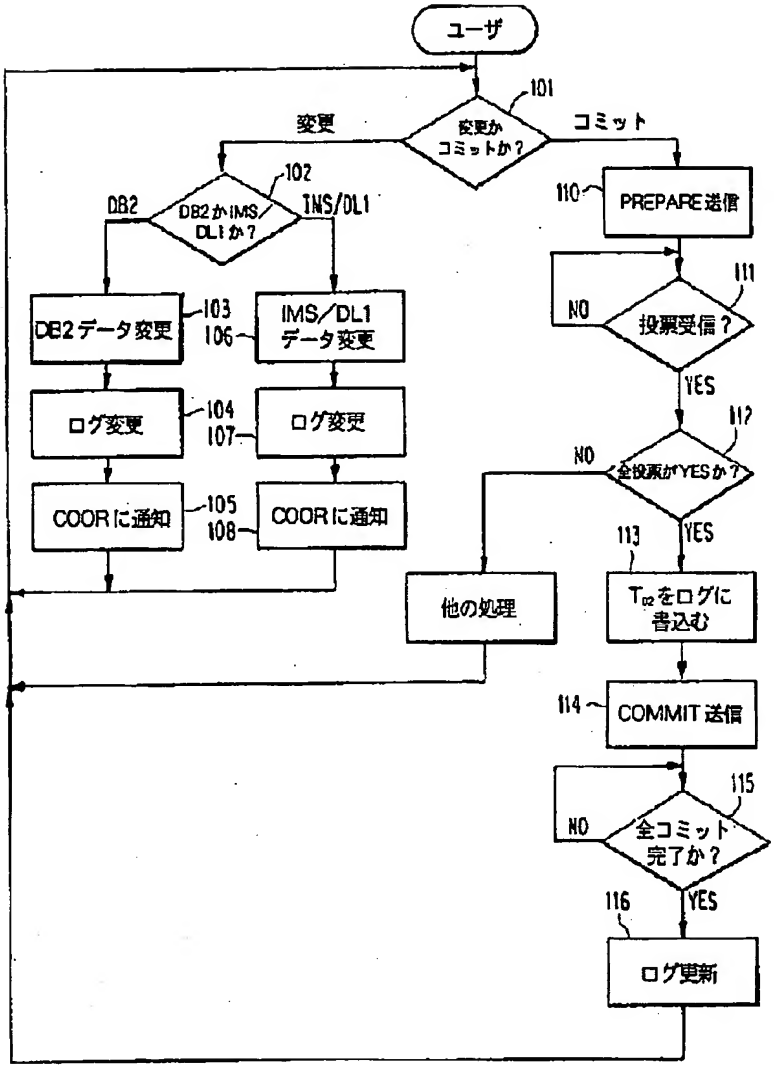
【図6】



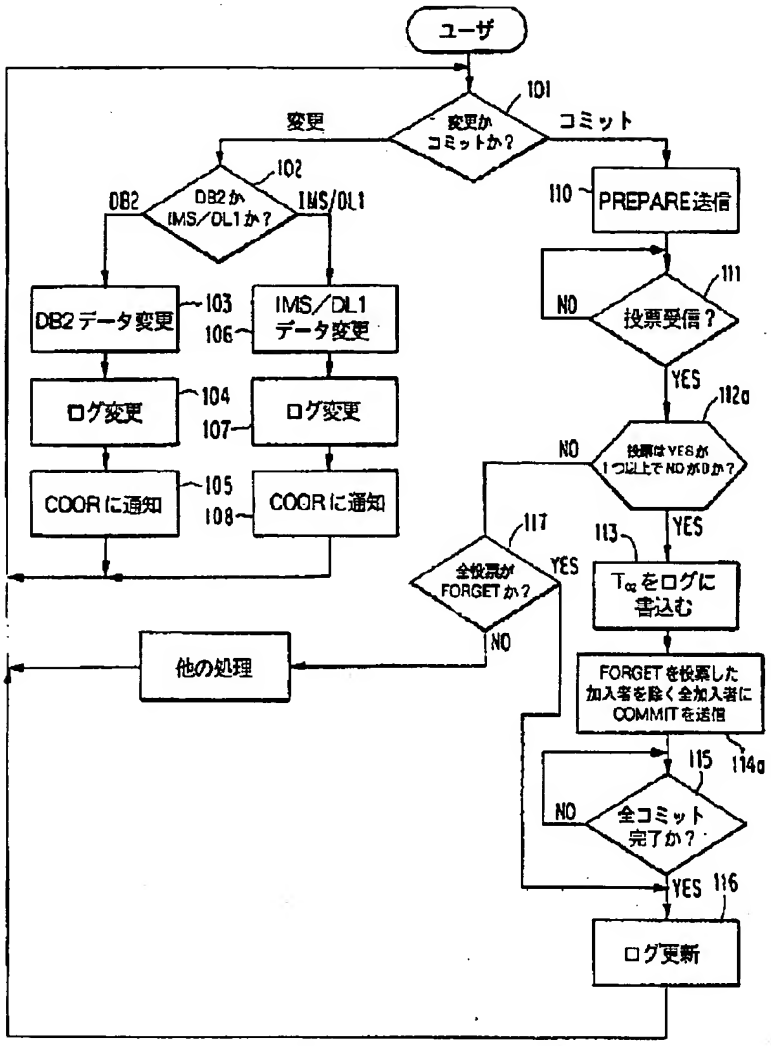
【図7】



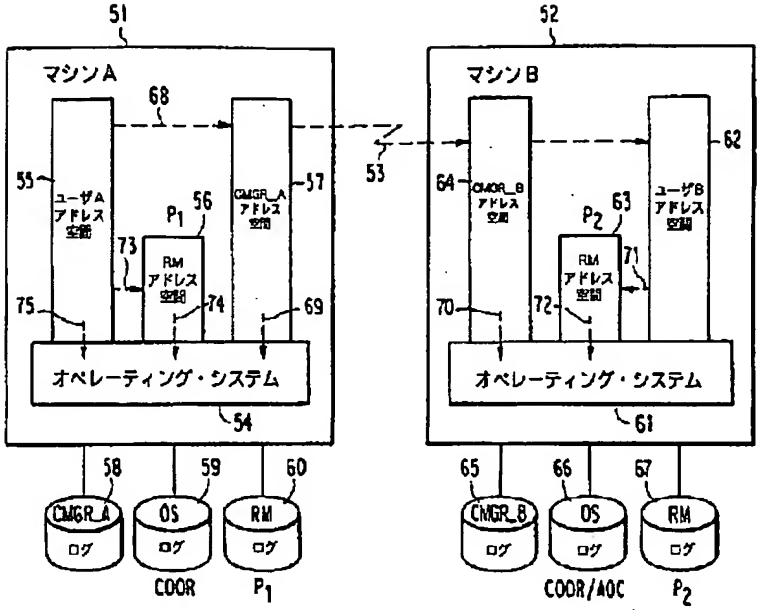
【図8】


ユーザ
101
変更か
コミットか？
変更
コミット
102
DB2かIMS/
DL1か？
DB2
IMS/DL1
110
PREPARE送信
111
NO
投票受信？
YES
112
NO
全投票がYESか？
YES
113
DB2データ変更
103
106
IMS/DL1
データ変更
ログ変更
104
107
ログ変更
COORに通知
105
108
COORに通知
他の処理
T_{02}をログに
書込む
114
COMMIT送信
115
NO
全コミット
完了か？
YES
116
ログ更新

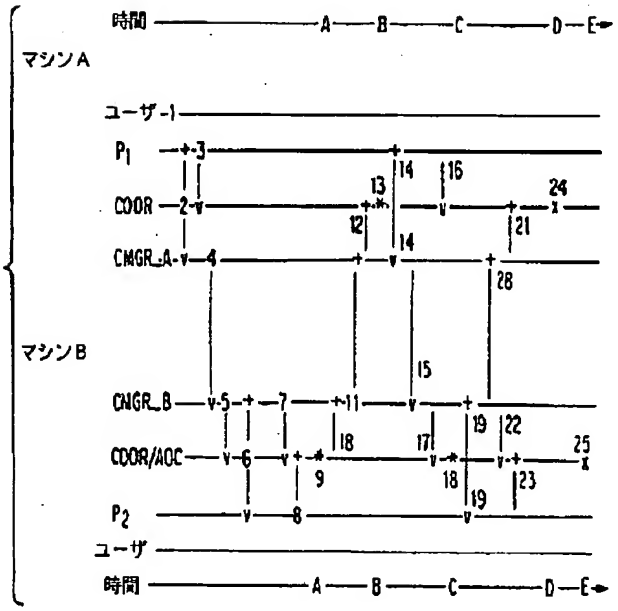
【図9】


ユーザ
101
変更か
コミットか？
変更
コミット
102
DB2か
IMS/DL1か？
DB2
IMS/DL1
DB2データ変更
103
IMS/DL1
データ変更
106
ログ変更
104
ログ変更
107
COORに通知
105
COORに通知
108
他の処理
110
PREPARE送信
111
投票受信？
NO
YES
112a
投票はYESが
1つ以上でNOが0か？
NO
YES
117
全投票が
FORGETか？
YES
NO
113
Tαをログに
書込む
FORGETを投票した
加入者を除く全加入者に
COMMITを送信
114a
115
全コミット
完了か？
NO
YES
116
ログ更新

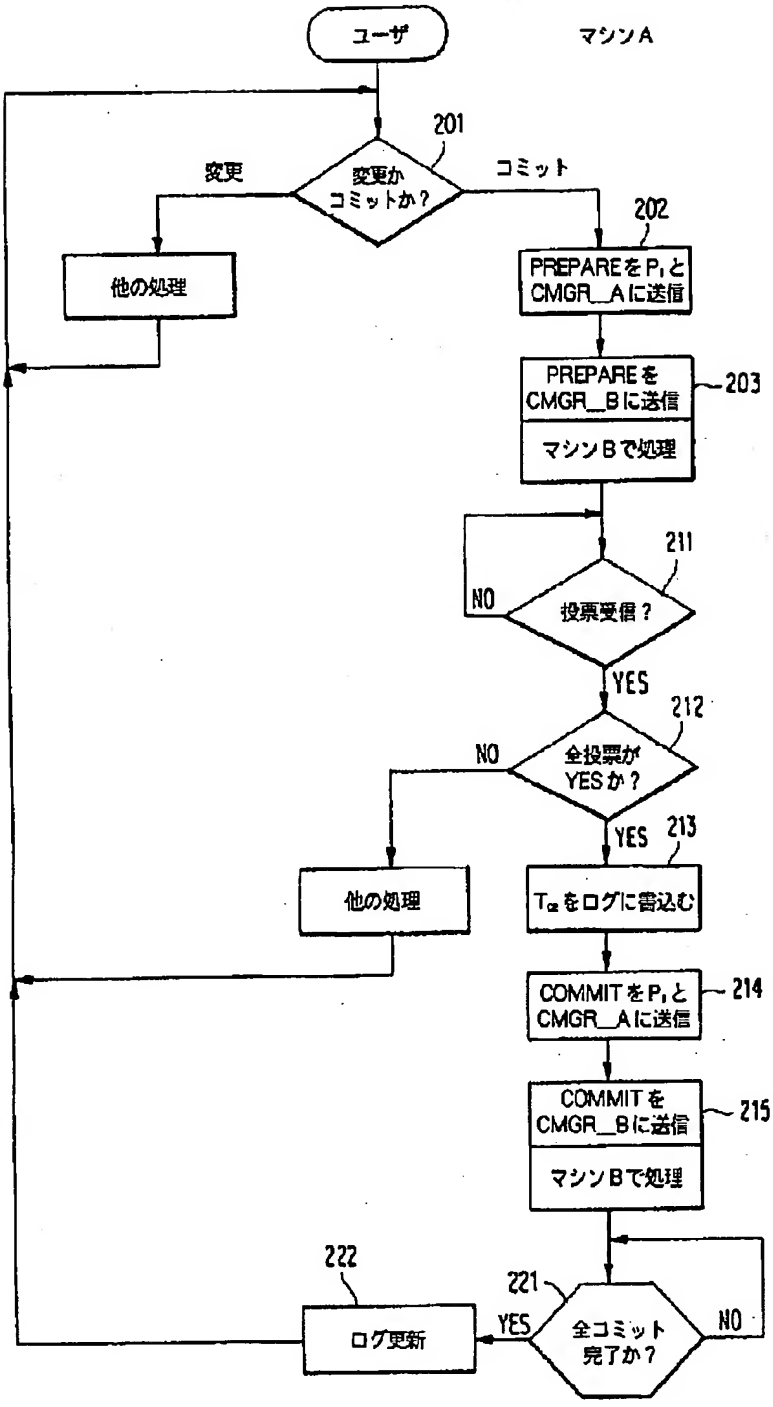
【図１０】


51
52
マシンA
マシンB
68
53
55
57
62
64
63
56
P1
P2
73
71
75
74
69
70
72
ユーザA アドレス空間
CMGR_A アドレス空間
CMGR_B アドレス空間
ユーザB アドレス空間
RM アドレス空間
オペレーティング・システム
54
61
58
59
60
65
66
67
CMGR_A
OS
RM
CMGR_B
ログ
COOR
P1
COOR/AOC
P2

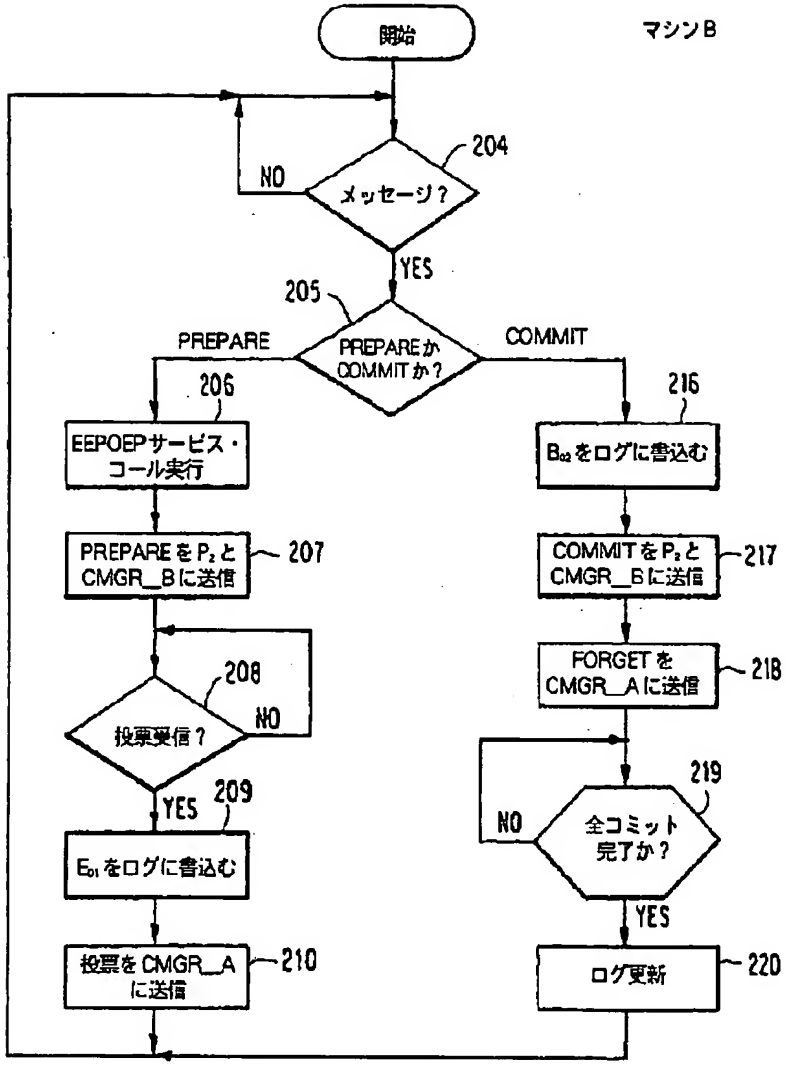
【図１１】


時間
A
B
C
D
E
マシンA
ユーザ-1
P1
COOR
CMGR_A
マシンB
CMGR_B
COOR/AOC
P2
ユーザ
1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
21
22
23
24
25
28

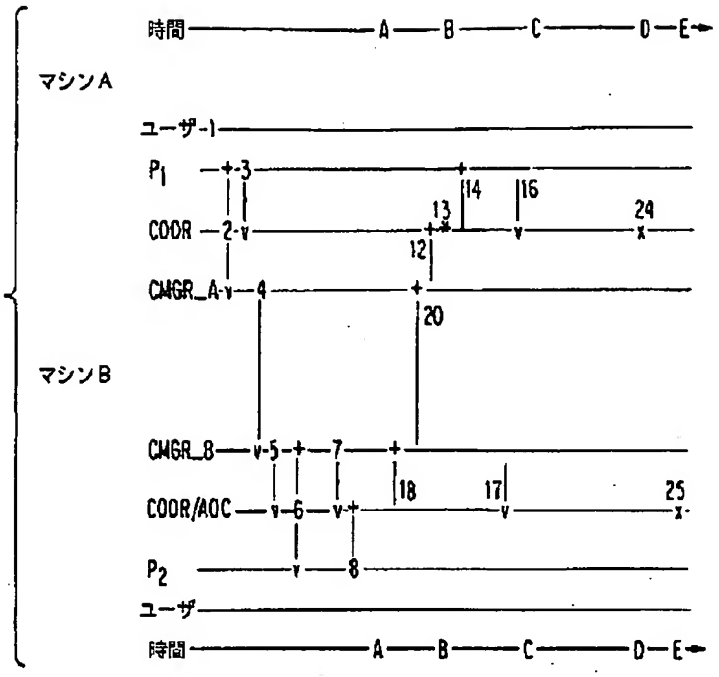
【図１２】


ユーザ
マシンA
201
変更か
コミットか？
変更
コミット
他の処理
202
PREPAREをP1と
CMGR_Aに送信
PREPAREを
CMGR_Bに送信
203
マシンBで処理
211
NO
投票受信？
YES
212
全投票が
YESか？
NO
YES
213
他の処理
Tαをログに書込む
COMMITをP1と
CMGR_Aに送信
214
COMMITを
CMGR_Bに送信
215
マシンBで処理
222
221
ログ更新
YES
全コミット
完了か？
NO

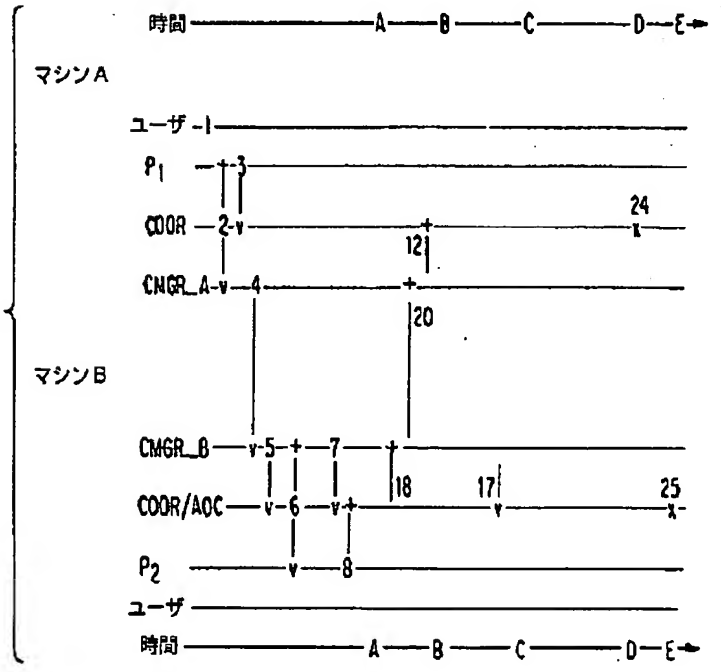
【図１３】


開始
マシンB
204
NO
メッセージ？
YES
205
PREPARE
PREPAREか
COMMITか？
COMMIT
206
216
EEPOEPサービス・
コール実行
B_{02}をログに書込む
PREPAREをP_2と
CMGR_Bに送信
207
COMMITをP_2と
CMGR_Bに送信
217
FORGETを
CMGR_Aに送信
218
208
NO
投票受信？
219
YES
209
NO
全コミット
完了か？
E_{01}をログに書込む
YES
投票をCMGR_A
に送信
210
ログ更新
220

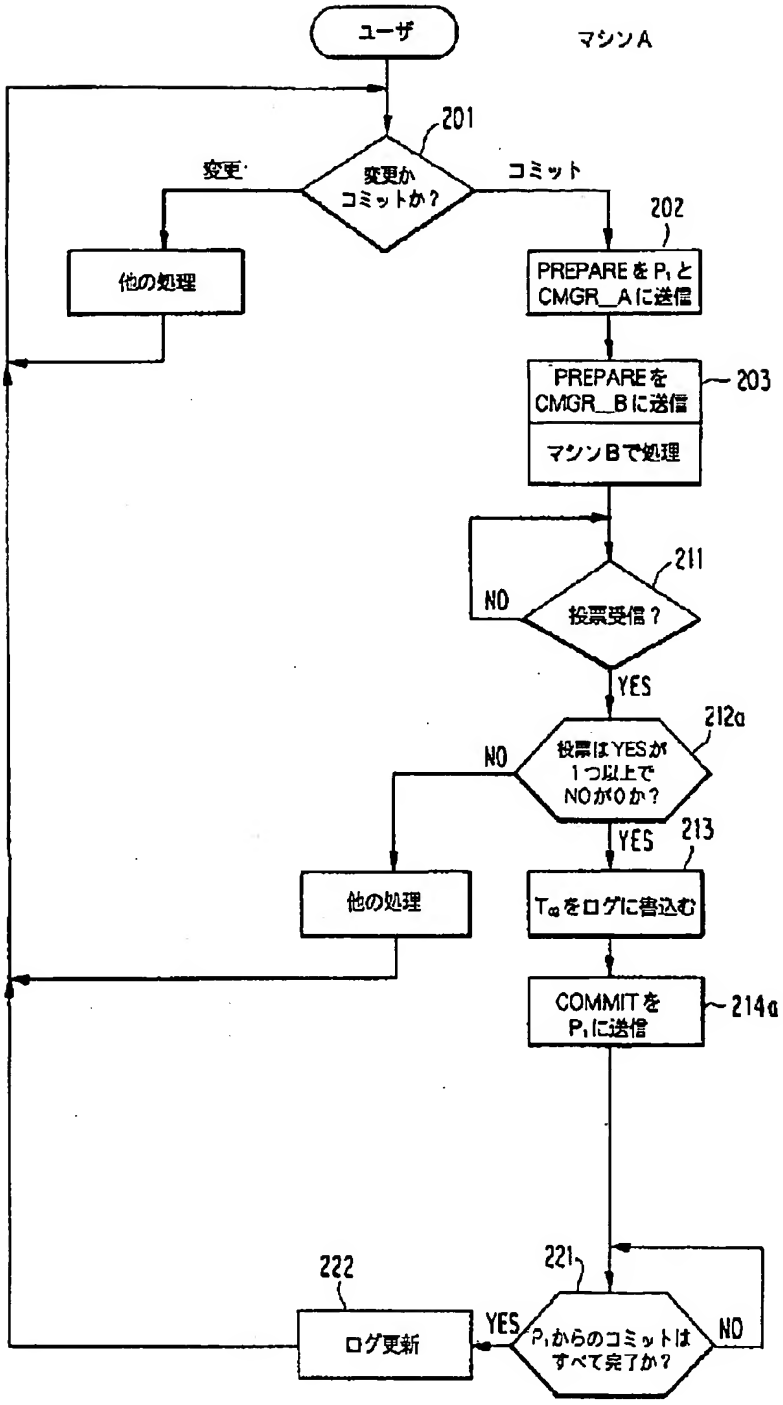
【図１４】


時間
A
B
C
D
E
マシンA
ユーザ-1
P1
COOR
CMGR_A
マシンB
CMGR_B
COOR/AOC
P2
ユーザ
時間

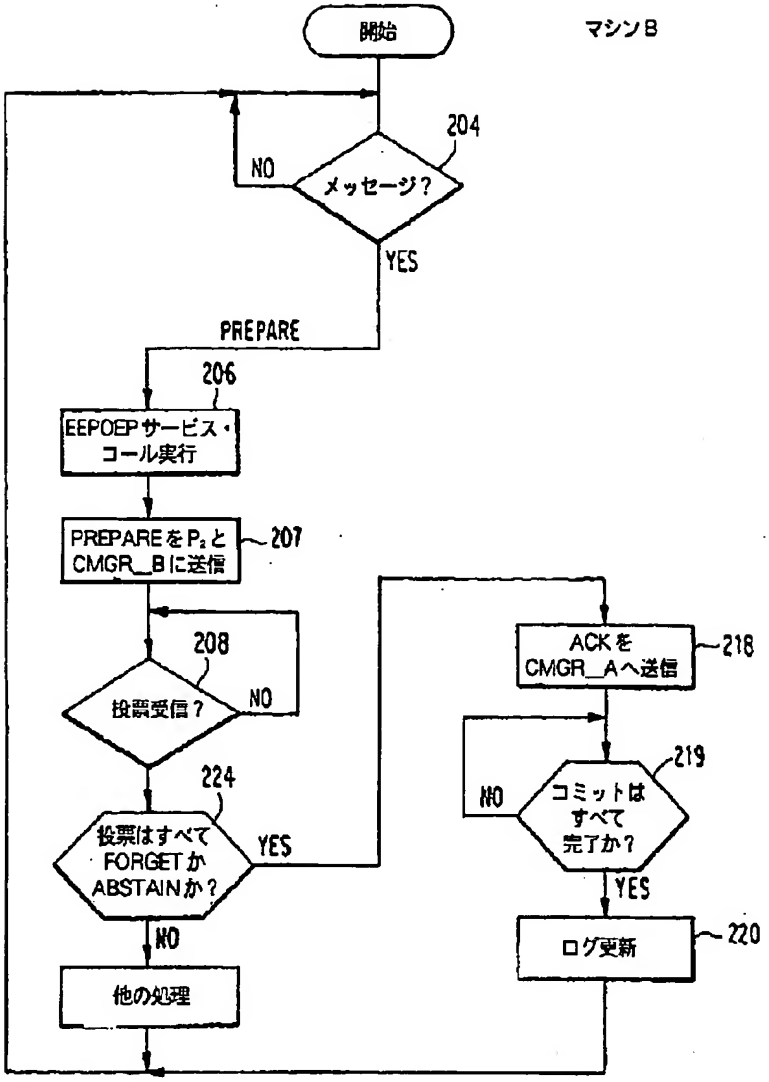
【図１５】


時間
A
B
C
D
E
マシンA
ユーザ-1
P1
COOR
CMGR_A
マシンB
CMGR_B
COOR/AOC
P2
ユーザ
時間

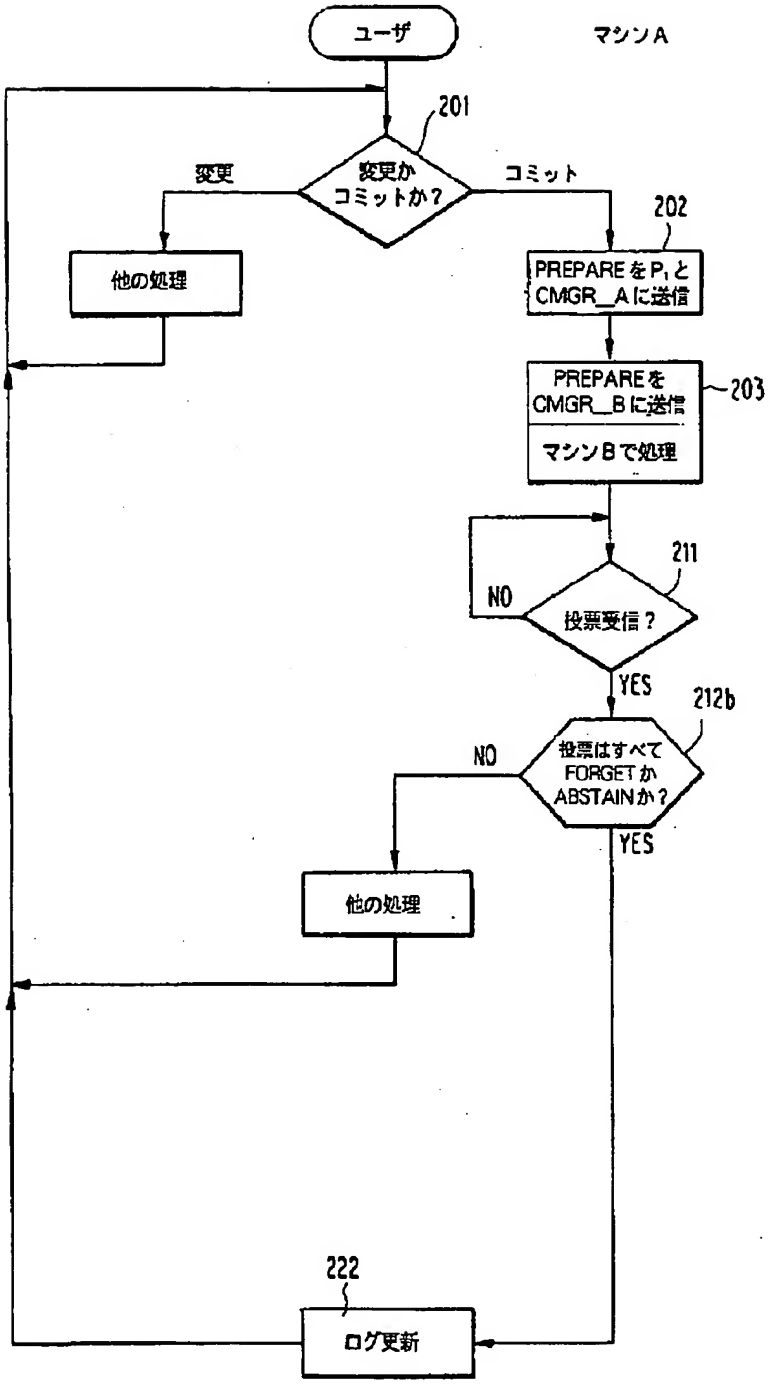
【図16】


ユーザ
マシンA
201
変更か
コミットか？
変更
コミット
202
他の処理
PREPAREをP₁と
CMGR_Aに送信
PREPAREを
CMGR_Bに送信
203
マシンBで処理
211
NO
投票受信？
YES
212a
投票はYESが
1つ以上で
NOが0か？
NO
YES
213
他の処理
Tαをログに書込む
COMMITを
P₁に送信
214a
222
221
ログ更新
YES
P₁からのコミットは
すべて完了か？
NO

【図17】


開始
マシンB
204
NO
メッセージ？
YES
PREPARE
206
EEPOEPサービス・コール実行
PREPAREをP₂とCMGR_Bに送信
207
208
投票受信？
NO
224
投票はすべてFORGETかABSTAINか？
YES
NO
他の処理
ACKをCMGR_Aへ送信
218
219
NO
コミットはすべて完了か？
YES
ログ更新
220

【図１８】



フロントページの続き

(72)発明者　リチャード・ダイベンドルフ
アメリカ合衆国94040、カリフォルニア州
マウンテン・ビュー、チャタム・ウェイ
345

(72)発明者　ダニエル・エドワード・ハウス
カナダ国、オンタリオ州スカボロ、ナンバ
ー1907、グレイストーン・ウォーク・ドラ
イブ　5

(72)発明者 アール・エイチ・ジェナー
アメリカ合衆国95135、カリフォルニア州
サン・ホセ、ガロウェイ・ドライブ 7688
(72)発明者 マーガレット・ケリー・ラベル
アメリカ合衆国12603、ニューヨーク州ポ
キプシ、フッカー・アベニュー 310
(72)発明者 マイケル・ジェラルド・モール
アメリカ合衆国12540、ニューヨーク州ラ
グレーンジビル、スクエア・ウッズ・ドラ
イブ 20
(72)発明者 スチュアート・ルイス・サイレン
アメリカ合衆国95037、カリフォルニア州
モーガン・ヒル、グリフィス・ウェイ
15630